A Belief System Model for Software Development: A Framework by Analogy

Paul Wernick

Thesis submitted for the degree of PhD

Department of Computer Science
University College London
Gower Street, London WC1E 6BT

April, 1996

"I ... realize that there is no special value to being, as it were, able to think. Indeed, one can see many disadvantages. The whole condition of mortals is created by their ability to analyze the universe and their inability to understand it."

(Corum of the Silver Hand, *in The Oak and the Ram*; Michael Moorcock; Quartet; London; 1974; pp 36-37)

Abstract

This work examines the belief system underlying computer-based systems development, by reference to an analogy with a model of scientific research due to Kuhn. Kuhn's model describes 'scientific communities', each united by an underlying many-faceted belief system, the 'disciplinary matrix', which forms a constellation of commitments shared by the members of these communities. A scientific community is compared here with the community of computer-based systems developers and its sub-groups. The division of the developers of computer-based systems development methods and tools into schools based on paradigmatic differences is paralleled with Kuhn's view of a scientific discipline at the early, pre-science, stage. The use of a computer-based systems development method in practice, and informal computer-based systems development activities, are paralleled with Kuhnian normal science, working within the paradigm of the discipline and of the techniques employed. This parallel provides a framework for structuring the explicit and implicit assumptions and models which form the craft knowledge underlying computer-based systems development theory and practice. Following a search for elements of the disciplinary matrix in the theory of computer-based systems development, as described in textbooks, and in its practice through interviews with developers, the results of action research and reports of systems development failures, it is concluded that the analogy with Kuhn's view of scientific activity is justifiable, and that articulation and examination of the implications of the analogy can reveal useful information to assist in describing and improving computer-based systems development. The results of this search are presented in terms of the specific beliefs and models identified. It is suggested that, as future research, the Kuhn-based model of computer-based systems development should be extended into a detailed investigation into the effects of individual elements of the disciplinary matrix, either individually or in combination, on the mind set of the computer-based systems developer.

Contents

Appendices

Tables

Acknowledgements

I would like to express my thanks to the following:

Celia Gould, for all manner of help, advice, and moral support during the years of work leading to this thesis, and for polishing the text of the thesis;

Russel Winder, my PhD supervisor; for advice, management, and controlling my wilder flights of intellectual fancy, and for showing the necessary patience and tact to get me to this stage;

Angela Sasse and Chris Clack, particularly for their conduct of the annual informal vivas, which really started me off, and kept me on, the right track;

Jeff Spearman, for listening to my wilder ideas on Software Engineering and keeping a straight face, and particularly for his vision of making software "appetising";

Prof. Paul Samet, for help and advice during the research reported in this thesis;

The anonymous computer-based systems developers interviewed for this work, who willingly gave their time and the benefit of their experience;

The anonymous referees of the paper 'A Plethora of Paradigms' (Wernick and Winder, 1993) for The Computer Journal, for forcing me to tighten up my ideas;

the Engineering and Physical Sciences Research Council (the then Science and Engineering Research Council) for providing a research studentship to support the work described in this thesis; and

The BBC Radio Test Match Special Team, for providing the best possible background for intellectual endeavours.

1 Introduction

1.1 Setting the Scene

1.1.1 The Problem Outlined

Peter Checkland has defined the word 'problem' for the purposes of his action research programme on soft systems thus:

> "A problem relating to real-world manifestations of human activity systems is a condition characterised by a sense of mismatch, which eludes precise definition, between what is perceived to be actuality and what is perceived might become actuality." (Checkland, 1990, p 155)

This condition well describes my feelings on being taught about current progress in 'Software Engineering' after over ten years' practical experience of computing as analyst, programmer, user support consultant, computer auditor and end user. Something was wrong.

The question of what was wrong began to resolve itself when I realised that all the differing methods[1] for software development were aimed at achieving the same end, i.e. delivering a (hopefully close to optimal) computer-based system to satisfied users. Therefore, in theory, all methods should exhibit many similarities. However, I noticed that a method's developers often seemed more interested in showing the differences between their method and those of others.

My objective has therefore been to determine whether an underlying philosophy can be found which:

- splits the theory and practice of *computer-based systems development* ("CBSD")[2] activities into that which unites all developers and that which divides them; and
- defines what is 'Software Engineering' and what is not;

---

1 I have used the term 'method' in preference to the more usual 'methodology' throughout this thesis, in order to preserve the latter term to describe 'the study of method' (and methods!).

2 This term is defined for the purpose of this thesis in chapter 2.

and to find out whether the derivation of the structure and elements of this philosophy can teach us something about the similarities and differences, strengths and weaknesses of different approaches to the problems of developing and maintaining computer-based systems. The work reported in this thesis is a first step in this research programme.

In the longer term, I hope that this work will allow us to:

- enable those software developers who currently do not use predefined methods, tools or techniques to identify and incorporate appropriate aspects of these in their work;
- compare existing CBSD methods, and identify strengths and weaknesses; and
- learn from other engineering disciplines, and apply methods and techniques from the latter to Software Engineering.

1.1.2 A Symptom of the Problem

Van Vliet (1993, p 215) quotes Rensch's (1982) vision of the future of object-oriented thinking from its earlier days:

> "My guess is that object-oriented programming will be in the 1980s what structured programming was in the 1970s. Everyone will be favour of it. Every manufacturer will promote his products as supporting it. Every manager will pay lip service to it. Every programmer will practice it (differently). And no one will know just what it is."

Rensch's prediction that definitions and usages of 'object-oriented' thinking will differ widely has certainly been confirmed by the intervening decade. However, I disagree with his suggestion that no one knows what 'object-oriented' means. Many people know. The problem is that they each 'know' one of the many different versions currently touted. As will be seen later, this is what might be expected from a Kuhnian pre-science discipline fragmented into schools, as is the talking at cross purposes and confusion which results. Rensch's prediction that "Everyone will be in favour of" object-oriented programming also fails to reflect the current fragmented state of CBSD theory and practice.

What is a new entrant into CBSD to make of the divergent definitions of what are now fundamental terms of art such as 'object-oriented'?[3] If they are taught by a person believing in one particular definition of the term, they may take that definition into other environments in which different definitions are held, causing a confusion which is the more dangerous for possibly not being recognised as such.

1.2 Scope of the Thesis

The work presented in this thesis relates to the design of software systems at all levels above that of the writing of program code; 'programming in the large' rather than 'programming in the small'. An initial investigation into the validity of the claim of 'Software Engineering' to be an engineering discipline, presented in Chapter 2 of this thesis, has suggested that the claims of the discipline to be 'engineering' in any rigourous sense are currently questionable. In setting the scope of my work, I have therefore ignored any such claims, together with the contentious matter of what is 'Software Engineering' and what is not.

The work presented here has been performed so as to refer to *all* CBSD work developing systems which are of any reasonable size; essentially, 'systems' rather than 'programs'. The exclusion of the lower size work levels has allowed the practice of CBSD to be considered in the context of a well-defined body of theory, embodied in the (too) many systems development methods.

The boundary of what is included in the scope of this thesis can be seen by reference to existing work on programming, such as that of Petre (1989). This was based on the results obtained by examining comparatively small programming examples; complete programs but not 'systems'. My interest is more in the work which needs to be done to produce practically useful *systems*, which implies the need for more than one program unit. To this end, I have examined a number of textbooks written to teach and/or inform computer-based systems developers, and conducted a series of in-depth interviews with individuals who develop or have developed computer-based systems for a living.

---

3 I view as speculative comments such as those made at an OOPSLA panel session (Russel Winder, personal communication, 1994) that the use of the term 'object-oriented' will converge over time, until such a convergence has been completed in both the theory and the practice of CBSD. I suggest that the academic and commercial pressures against such a unification are at present considerable.

In some areas, the boundary of my work has not been clear-cut. For instance, those among the in-depth interviewees who develop computer-based systems by themselves do not appear to take a decision at some point to stop 'developing systems' and start 'programming'. This required some judgement on my part in analysing their interview transcripts, since some of their comments are described in programming (module level) terms, but reveal beliefs which are valid at the systems level (or sometimes both levels, e.g. documentation).

The scope of the work set out in this thesis is as broad as possible within the context of 'programming in the large'. The concepts and theories set out in it are defined in such a form as to cover:

- all sizes of computer-based systems development, from the smallest involving more than one programming unit to the largest system;
- all types of computer-based systems, designed for any purpose or environment, rather than being restricted to 'information systems' on which much of the related work (see Chapter 8 below) has been carried out; and
- all phases of the life of a computer-based system, from initial analysis to the decision to end maintenance – whether these are years apart for a large computer-based system, or hours apart for a file conversion utility to be used once and then thrown away.

In terms of the people involved in developing computer-based systems, the scope of the work comprises:

- those who develop and maintain the software and intellectual tools used to develop computer-based systems;
- those who develop and maintain computer-based systems;
- those who teach developers and maintainers of computer-based systems; and
- those who manage these activities.

1.3 Claims Made in the Thesis

1.3.1 In General Terms

The major claims made in this thesis are that:

- Kuhn's model of science is a reasonable analogy for CBSD;
- Kuhn's model of science is a useful analogy for CBSD, in that it helps explain the current state of the discipline and provides pointers to possible routes to future improvements;
- there exists a complex system of beliefs, models, values, etc. – the disciplinary matrix ("DM"), or paradigm – of CBSD;
- some of the elements of the DM can be identified by an examination of current theory and practice; and
- the elements forming the DM have an effect on the work of the software developer.

1.3.2 The Main Argument

Software Engineering has not yet fulfilled the expectations raised in the public community by its proponents. Neither has a consensus behind a single mechanism or set of mechanisms for the development of software been achieved between software practitioners, despite the undoubted progress in CBSD methods since the Garmisch conference on Software Engineering (Naur and Randell, 1969). There is currently no sign of such a consensus emerging, and the rate of introduction of new mechanisms and techniques presents a picture of divergence rather than convergence in CBSD practice. This turmoil can be contrasted with the position of other design disciplines, in which it appears that there is general agreement either on the mechanisms needed to support design activities, or on the validity in general terms of the options available – see, for example, the calculation-based approach to electrical engineering design in Shepherd *et al.* (1991).

I suggest that this situation is due, at least in part, to the lack of any well-understood belief system underlying current CBSD theory and practice. The consequent inability of CBSD theorists and practitioners to understand the influences of the discipline's mechanisms on its activities demonstrates that the

status of Software Engineering should be considered for the present to be that of a craft rather than an engineering discipline, despite its name[4].

The same argument might explain the failure of CBSD tools and techniques to influence the individual or informal developer. Since there are many different mechanisms, all seeming to try to do the same thing, and all claiming success in their different way, how can a developer who cannot afford the time or money to go on many courses or read many books learn enough about them to make a rational choice? We can contrast here the considerable influence of modern implementation languages and tools, such as C++, screen building libraries and so on, which can be seen by the individual developer to provide an immediate benefit, on those individual developers, with the lack of influence of new design notations and methods, whose influence is indirect and benefits are unproven.

One consequence of the lack of a theoretical basis for Software Engineering seems to be the failure to have achieved an understanding of the effects of currently used and proposed mechanisms of CBSD on the products of software development. Implicitly, this statement is a criticism of current Software Engineering research, insofar as it appears to be ignoring this issue. Current work which considers the need to trade off the influence of the thinking behind particular types of methods (for example Gasson, 1994) does not link to any consideration as to how effectively the influences are reflected in individual methods.

It is postulated that the use of any predefined mechanisms during a particular CBSD process produces a definite effect on the results of that process. The effects of an underlying belief system on the process and products of CBSD is considered in the context of Kuhn's theories on the mechanism of scientific progress. However, any resultant identification of Software Engineering as a science is explicitly denied. The identification of elements of the DMs of CBSD methods, taken with equivalent work seeking to identify elements underlying informal CBSD mechanisms used by individual developers, is taken to be a sensible first step towards understanding the effects of the mechanisms used for CBSD on the products of that development.

---

4 The over-selling of the possibilities of software, given the current state of the art, is also a part of the problem of unfulfilled expectations for software users.

It is suggested that, by using Kuhn's concept of the DM as a structure within which to work, it may be possible to identify specific elements in the set of beliefs which guides CBSD. This can also lead to consideration both of the results of these beliefs, whether implicit or explicit, and of possible alternative beliefs and the effects on the resultant software production. A structure is described which divides the existing identifiable belief system elements into those which are accepted by all computer-based systems developers and those which divide practitioners in the discipline into schools. Some examples of the individual beliefs, models, values and so on forming the elements of this DM are presented, but the work of confirming these, identifying further elements and determining the practical effect of the elements on computer-based systems development or on each other[5], either alone or in combination, is outside the scope of the thesis, with the exception of one example examined to test the validity of the thinking underlying the thesis.

1.3.3 The Relative Importance of the Concepts Introduced in this Thesis

The concepts introduced in this thesis are:

- a model of the current state of CBSD based on Kuhn's philosophy of science;
- the use of the DM as a framework for setting out and considering the implicit and explicit influences on CBSD theory and practice; and
- the specific elements of that DM listed in Appendix 2, and the Algorithmic Model in Chapter 7 describing how they might be visualised as combining to form the complete DM affecting a CBSD activity.

I do not regard these three as being of equal importance for the argument of the thesis. I see the DM, as a concept, as the most important, since it has the potential to change the way in which CBSD is considered by its theorists and practitioners. It may provide a means for changing the ground on which the debate over how to develop software is fought, from that of the explicit attributes of competing methods (process model, modelling techniques, etc.) to that of the beliefs and

---

5 The adoption of some elements may explicitly deny the adoption of others. Such connections may imply that the elements concerned are connected by virtue of being different expressions of the same problem, such as mechanistic vs. user-centred viewpoints of the users of a system, and the different expressions found here for basically the same problem are as would be expected by Kuhn on the basis of his comments on incommensurability.

assumptions underlying those attributes, and eventually towards consideration of the effects of both the attributes and the underlying beliefs on software systems.

Second in importance is the model of the current state of Software Engineering. Its main utility is in explaining many of the current concerns expressed in research into CBSD, and in providing a mechanism by which the research communities within CBSD can be given an understanding of the real issues over which they currently seem to be battling.

Finally, and least important for the argument of the thesis, are the individual elements listed in Appendix 2, and the Algorithmic Model described in Chapter 7. Whilst I believe these are valid, and as well supported as possible within the scope of this thesis, they are only the first steps towards setting out the complete DM underlying CBSD. In particular, the work on determining how DM elements interact, and how they affect software developers, is as yet unstated. This forms a considerable part of the future work outlined in Chapter 9. However, the search for elements has allowed me to begin exploring the structure of the DM below the broad categorisation made by Kuhn. This structure is outlined in Chapter 6.

I believe that the evidence presented in this thesis for the existence and effect of the DM, and for the goodness of fit of the Kuhn-based model of a discipline, in CBSD are fairly strong. However, the validity of the concept in this area must still be regarded as 'not yet falsified'. The individual elements are each much less well supported to date, but the future work described in the concluding chapter indicates ways to examine them for such support. The aim of the work performed to find individual elements has been designed only to:

- provide some evidence for the existence of elements of the DM which had been postulated at an earlier stage;
- demonstrate the feasibility of different ways of looking for these elements, specifically trawling through literature to find the theoretical ideas, and interviewing practitioners to see what actually happens in the real world; and
- provide some evidence to support the predicted existence and effects of the DM as a whole.

1.4 Contributions of the Thesis

I describe here in outline the contributions made by the work presented in this thesis. Since the work reported here is the first step in a potentially large research programme, many of the contributions are pointers to future work, and I refer the reader to the Future Work section in Chapter 9 for a more detailed discussion of how these objectives might be achieved.

1.4.1 Short-Term Gains

The contributions of the work reported in this thesis can be summarised as follows:

- providing both a framework and a rationale for consideration of the influences on computer-based systems developers of current and future theory and practice;
- providing evidence to support this framework and the identification of some of the elements of which it is composed;
- starting the process of setting out the implicit influences which have not previously been considered explicitly in designing CBSD methods, and which can now be laid open to examination and possibly rejection;
- providing a mechanism for future work in tracing the history of CBSD, in terms of changing sets of underlying beliefs held by software and technique developers, which will provide a clearer explanation of this history; and
- showing weaknesses in the current state of Software Engineering, and suggesting reasons for these weaknesses.

In addition, this work promises to provide a basis for:

- helping define Software Engineering, to refine the body of knowledge needed by software engineers and thus to help set out a syllabus for what the professional software engineer should know;
- defining who is a 'software engineer' by reference to an underlying belief system common to all 'software engineers';
- illuminating and documenting fundamental differences between the roles involved in CBSD activities, by reference not to job descriptions, i.e. what

people do, as at present, but by reference to what they need to know and how this knowledge affects the ways in which they go about their work;

- informing the process of selecting and tailoring methods for specific applications, by relating the DM of the method to the problem, and based on the known effects of the DM;
- helping to build new or modified CBSD methods by understanding the effects of method design decisions on method use;
- assisting in educating new software engineers, particularly by showing them the effects of method choice;
- providing support for socio-technical research into CBSD, such as the Collective Learning work at University College London (Winder *et al.*, 1992);
- identifying parallel disciplines from which to learn, by identifying common DM elements between disciplines once there are documented DMs for both (and the possibility of such comparisons could provide a spur to investigate these DMs); and
- assisting in the development of definitions of key terms which can be agreed by all computer-based systems developers, by bringing out differing assumptions implicit in the current controversial definitions.

The additional work needed to address these objectives is considered in the Future Work section of Chapter 9.

1.4.2 Placing this Work in the Context of an Envisaged Research Programme: The Long-Term Objective

The eventual objective of the work which I outline in this thesis, the vision which I have of the development of Software Engineering and of CBSD in general, is that of a discipline able to deploy a theory of the application of its predefined mechanisms to CBSD which will allow the effects of such application to be determined in advance, at least in broad, general terms. It will be an engineering discipline in the terms of Long and Dowell (Dowell and Long, 1989; Long and Dowell, 1989), with an underlying philosophical and science base.

The successful achievement of this long-term goal is dependent on the following assumption; that:

The use of a predefined mechanism in the development of a software system will tend to result in the characteristics of that system being qualitatively affected in a manner which can to some extent be predicted from the belief system underlying that predefined mechanism.[6]

A long-term objective of the work outlined and commenced in this thesis is to determine the dimensions of this belief system and the effect of each dimension both alone and in combination with others. The thesis sets out a structure in which these dimensions could be placed on the basis of an analogy with one strand in the Philosophy of Science.

The claim to any possible predictive power of the theory presented here is currently limited to qualitative estimates, since we are dealing with the development of software in a human-based environment of developers and users, using human creativity, imagination and design skills as the basic tools. These claims are based on suggestions as to how the theory in this thesis might be applied rather than as the direct results of experimental research.

The work presented here has been carried out without reference to psychological or sociological theories, being based purely on the phenomena encountered in the theory and practice of CBSD. Attempts to provide a quantitative measure of the effects of belief systems on such activities might be based on the application of sociological and psychological research.

1.4.3 A Long-Term View of Computer-based Systems Development Practice

In the long term, an approach to CBSD might be envisaged, employing as a pro-active tool the understanding of the effects on a computer-based system of the mechanisms used to develop that system.

The mechanism will need to perform the following activities:

- determine the required attributes of the system to be developed;

---

6 This assumption is examined later in the context of both the work performed for this thesis and that of other researchers.

- select a set of beliefs, models, values and so on which will influence developers in favour of these attributes[7], perhaps incorporating equivalent beliefs relating to organisational factors as these beliefs are identified, and taking into account the interactions between the beliefs forming the set;
- select *tested optimal* design features for a method which encompass these beliefs, taking into account the interactions between the design features; and
- select, modify or build an application-instance-specific method which incorporates these features.

It is important to note that, in order to be able to use this mechanism, it is necessary that computer-based systems developers be trained to:

- look for and identify the desired attributes of a system;
- understand how these attributes relate to the individual elements of a method's belief system;
- relate these elements to each other to build a belief system which is consistent and supports a method;
- build a consistent and usable method which reflects the required beliefs; and
- keep all decisions under constant review bearing in mind the inductive nature of Software Engineering.

In addition, the theory set out in this thesis will need to be operationalised to provide the information to apply it in this practical manner. The requirement to acquire the necessary skills will require a new approach to Software Engineering education, emphasising these new skills as well as those needed directly for systems analysis and design.

How might the activities described above actually be fitted into a real world situation? From the point of view of the systems development organisation, it may

---

7 For example, is innovation seen as important? If so, given that, as suggested later in this thesis, development process models without feedback may tend to stifle innovation, then a process model allowing feedback should be considered, albeit in the light of the effects of the other factors needing to be taken into account.

be necessary to maintain a current model of how they achieve their objectives, i.e. developing systems. This model may be in the form of conceptual models similar to those employed in the Soft Systems Methodology (Checkland, 1990). Once a client's perception, either of a problem or of the need for a system, has been examined to discover the beliefs required by the system developers and a suitable DM defined, the effect of this on the developers' conceptual model will have to be considered and that model modified as required to allow the successful use of the method which will be built from the DM. Over time, a set of successful organisational models and methods may be built up; these will encapsulate the development organisation's experience in a more formalised way than can occur at present.

The approach proposed here is similar to that of current method tailoring, such as that in SSADM v.4 (Downs *et al*, 1992), and loose framework-based approaches (for example Multiview – Avison and Wood-Harper, 1993), but differs in the greater degree of formalisation offered by a well-understood DM allowing reasoned decisions to be made, and in the completeness of freedom of choice given to practitioners among the available tools. Any tool whose underlying belief system has been analysed can be considered since its fit to the requirements and the other tools already selected can be checked. The mechanism proposed here also has some similarities to that of van Vliet (1993, pp 222–224) and Episkopou and Wood-Harper (1986) when they propose guidelines for selecting a CBSD method for a project, but operates at a finer grain than either of these, both in the criteria for selection and the tools and/or techniques to be employed.

This simplistic description of how a theory based on the work presented here might work ignores some aspects of the real world, such as the effects of individual developers' belief systems on CBSD, the cost of training and retaining staff in the use of new methods, and the potential advantages of having staff experienced in the method to be used. However, as a vision of the future use of CBSD methods, it provides a mechanism for the modification of CBSD methods which has a firmer theoretical base than the current *ad hoc* mechanisms. The essential difference is that the process starts with determining the attributes of the system to be built, and develops a method to match these attributes. The first step in systems development becomes a combination of problem analysis and method development. The process for method development may even be found to be similar to that for systems development, requiring analysis of organisational and system-specific attributes, perhaps by means of a soft method of some sort.

If the benefits of such an approach are found to be sufficient for their purposes, it may also become worthwhile for informal developers to adopt it, or a cut-down version of it, either to replace the total informality found in some of the subjects interviewed for the work presented here, or to support their own selection of formalised systems development techniques or tools. The mechanism therefore provides a possibility of uniting the working practices of formal and informal software developers to a much greater degree than currently appears to be the case. The modification of formalised methods for less formal use is already occurring; a real-world example of this was given by one subject during the interview work presented in this thesis. However this mechanism is currently as informal as is much of the work towards the development of new methods. This area of method tailoring therefore provides an example of an area of current work which might be informed and improved by an extension of the work presented in this thesis.

In summary, the completed DM of CBSD, whose structure and content are outlined in this thesis, will allow the experience of computer-based systems development practitioners to be preserved and applied in a more rational way than is presently possible.

1.5 Structure of the Thesis

The structure of the rest of this thesis is as follows:

- *chapter 2:* a critique of the current state of the theory and practice of computer-based systems development, to bring the issues and problems into focus;
- *chapter 3:* a short description of approaches in the Philosophy of Science, and an examination of the work of Kuhn in particular, including some criticisms made of his work, to provide a background for the arguments introduced in Chapters 4–8;
- *chapter 4:* an initial consideration of how Kuhn's work can be applied to CBSD, to place the current state of the discipline in the context of his models of a discipline;

- *chapter 5:* a description of a series of informal investigations into the theory, practice and results of CBSD, testing and exploring in CBSD activities the concepts introduced from Kuhn;
- *chapter 6:* a consideration of how the analysis of CBSD can be extended as a result of both the investigations and of further theorising, demonstrating the power of the Kuhn-based model of CBSD in explaining observed phenomena and as a reasoning tool;
- *chapter 7:* a conceptualisation of how a set of beliefs might be built into the DM which influences a particular systems developer in a particular situation, to assist in understanding the relationship of the theory described here to instances of the practice of CBSD;
- *chapter 8:* a review of the work of other researchers which is relevant to that described here, showing how my work relates to existing strands of published research; and
- *chapter 9:* a summary, conclusions and a description of future work which might be performed to extend and explore the theory presented in this thesis.

In addition, the questionnaire used to structure the interviews with computer-based systems developers, and the list of elements of the DM of CBSD identified during the work reported here, indicating the sources from which I have derived them, are included as Appendices 1 and 2 respectively. A list of the abbreviations used in this thesis forms Appendix 3.

2 The Current Position of Computer-based Systems Development

2.1 Introduction

My experience and investigations have led me to conclude that there are problems with Software Engineering as currently theorised and practiced. These include:

- an observable mismatch between Software Engineering theory (as expressed in textbooks) and practice (as performed in practice) – see, for example Curtis *et al.*, (1988, p 1283);
- highly-publicised examples of failures of high-profile CBSD projects, such as the London Ambulance Service command and control system (LAS, 1993), and continuing doubts expressed concerning safety-critical systems such as those which control airliners[8] and nuclear reactors, and the differing focus of critics on why these problems have arisen;[9] and
- disagreements as to how to define the term and the discipline 'Software Engineering'.

In this chapter, I seek to define the discipline to which the work in this thesis relates, and then to determine whether it is achieving the level of success to which it might aspire.

2.2 Defining Terms

As a preliminary to detailed analysis of the discipline whose current state I will shortly criticise, it is useful to define the terms used to describe that discipline. I therefore delineate, and distinguish between, 'Software Engineering', 'software development' and 'computer-based systems development'.

---

8 Funk *et al.* (1995) refer to three recent major accidents, involving two Airbuses and an ATR-72.

9 The reasons for failures are usually seen as being either managerial or technical; contrast the views of Flowers (1994) and the official report (LAS, 1993) on the reasons for the failures of the London Ambulance Service computerisation; the former sees the problem as technical, whereas the latter refers to managerial failures as the main problem.

2.2.1 Software Engineering

The term 'Software Engineering' was first used by the conveners of the conference on the software crisis held at Garmisch in 1968 (Ralston and Reilly, 1993, p 1218). Naur and Randell (1969, p 13) noted at the time that:

> "The phrase 'Software Engineering' was deliberately chosen as being provocative, in implying the need for software manufacture to be based on the type of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering."

Since then, it seems that the term has been used almost as a weapon, to press the claims of the discipline of software systems development to be an engineering discipline irrespective of its actual status. There is an implication that some aspect of 'quality' is associated with the products of a 'Software Engineering' process.

More recent attempts to define the term 'Software Engineering' have resulted in some interesting if not necessarily useful consequences. For example, Sommerville (1992, p 2) notes that:

> "There are a number of possible definitions of software engineering. Their common factors are that software engineering is concerned with software systems built by teams rather than individuals, uses engineering principles in the development of those systems and includes both technical and non-technical aspects."

I am concerned about two of the three parts of this definition:

- I see no reason why the definition should exclude the development of software by individuals. If, for example, an individual uses a predefined method in a correct manner or a formal method according to the instructions, how is it to be said that this is not Software Engineering? When Sommerville concludes that "Software Engineering" as he defines it is an activity performed by *groups* of people, he explicitly demotes those who develop software by themselves to a lower form of work, since they *cannot* be doing 'engineering'.
- What are the 'engineering principles' used in Software Engineering? I believe that this thesis is an example of a first attempt to establish the

> underlying belief structure of computer-based systems development, that the general engineering principles, if any, of the discipline are yet to be determined, and that the previous lack of a theory such as that presented here makes very difficult the application of true engineering principles to the development of highly complicated artifacts with differing requirements under diverse circumstances.

It seems to me that Sommerville's summary and the book which follows, can be said to have as much to do with the idea of the engineering of the *software process* as the *products* of that process.

Another definition of 'Software Engineering' seems to be concentrating more on the *products* to the exclusion of the *process*:

> "Software Engineering is that form of engineering that applies the principles of computer science and mathematics to achieving cost-effective solutions to software problems". (Ralston and Reilly, 1993, p 1218)

There is no consideration here of any need or desire to control the *process* of developing software, other than any support which "the principles of computer science and mathematics" can give to controlling costs.

I suggest that the term 'Software Engineering' is now perhaps best applied either to the theoretical side of computer systems development, such as the development of software development methods, or to the transformation of sets of requirements, recognised by mechanisms outside Software Engineering, into a form, executable on a computer, which conforms to those requirements. Its use as a description of software development in the large is still problematical and engenders hopes which have not yet been fulfilled, as shall be seen below. More accurately descriptive terms for what is currently called 'Software Engineering' might perhaps be 'Computer Systems Development' or 'Information Systems Development'.

2.2.2 Software Development

This term encompasses all work which is mainly concerned with writing, and possibly maintaining, software. No statement is implied as to the size of the software artifacts, nor the quality of the results. I therefore view 'software development' as including programming-in-the-small as well as programming-in-the large.

'Software development' encompasses all such activities from the smallest with limited success criteria such as the writing of utility programs to be used once and then thrown away, to the development of large safety-critical computer-based systems.

2.2.3 Computer-based Systems Development

My preferred term for large-scale computer system development is 'Computer-based Systems Development' ("CBSD"). This is the basis of the scope of the work presented in this thesis.

This term is intended to cover all work of some size larger than a 'program', but is deliberately formulated so as not to imply any statement as to the quality of the result, nor of how this is achieved. Whilst the term 'Software Engineering' may seem perhaps to imply the use of a predefined CBSD method, the application of formal methods or the use of a CASE tool, CBSD can happily proceed in the absence of any of these.

Additionally, the term CBSD is intended to recognise that many computer-based systems are built with some human interaction in mind. The task of CBSD must include the development of tools to handle such interactions, extending to the design of complete work-systems and to the consideration of the organisational implications of implementation. This is neither implicit or explicit in the label 'Software Engineering'.

Systems which do not have a large software component have also been excluded from the work in this thesis. This is because of the availability of a large identifiable body of published information on how software systems developers should go about their work, and because of the broad limitations on scope set by a PhD programme.

I therefore see the term 'computer-based systems development' as being broader in scope than 'Software Engineering' truly applied, since it may include non-engineering approaches. In comparison with 'software development', it is both narrower since it excludes the development of any software artifact not of a size which can reasonably be regarded as a 'system', and wider since it includes the broader context of the deployment of computer-based systems.

2.2.4 Summary

I identify three terms, reflecting different viewpoints towards the development of software-based systems:

- *Software Engineering*, which is a term used to give legitimacy to the development of large software systems, usually by teams of people;
- *software development*, which I define as encompassing the development of all software artifacts; and
- *computer-based systems development*, which covers the development of large systems incorporating a major software component, but including the non-computer-based aspects and without any indication that any particular number of people is involved in the work.

The last of these three terms, computer-based systems development ("CBSD"), forms the basis for the work presented here.

2.3 Scale and the Problem Domain

I do not intend to consider further the problems of programming, i.e. smaller software development activities. These and the problems of CBSD, although they may overlap, are perceived as being different due to the *scale* of the problem involved.

This difference was noted at the Garmisch conference in 1968, where problems were documented which were believed to be due to trying to scale up small-scale software design mechanisms to meet large-scale problems. Perlis, in the conference's keynote address, referred to "...poor performance, poor design, instability and mismatching of promise and performance..." which had arisen due

to "...a change of scale which we do not yet know how to reduce to alphabetic proportions." (Naur and Randell, 1969, p 135). David also refers to the 'scale' which caused problems with developing software as comprising both the well-recognised idea of large numbers of lines of code, and other elements such as "...the number of different non-identical situations which the software must fit ... the different hardware configurations which the software must accommodate, and ... the range of input error conditions which it must accommodate gracefully. I'm sure you can think of many more." (Naur and Randell 1969, pp 68–69).

Sommerville's more recent definition quoted above suggests that Software Engineering is related to problems of a size requiring *teams* of developers to solve them. As will be seen later, the belief that the size of the problem affects the way in which it is to be addressed is still current among CBSD theorists and practitioners.

Since scale seems to be an aspect which affects the mechanisms used for software development, I intend not to examine the problems and issues of smaller scale programming work but to confine my work to software development in the large.

2.4 A Critique of Software Engineering

In this thesis I intend to look specifically at the problems and issues of CBSD. However, a critique of the current state of Software Engineering forms a useful commentary on the current state of work of those people who are making an effort to 'do it right'. How well has Software Engineering met the challenges set at Garmisch and subsequently? Does it live up to the hype explicit in the label?

To answer these questions, I have examined the current state of Software Engineering in the light of a model of disciplines devised by Long and Dowell (1989).

2.4.1 Dowell and Long's Characterisation of a Discipline

2.4.1.1 Conceptions

As part of their work to characterise current Human Factors practice,[10] Long and Dowell (1989) have described the concept of a 'conception', by which term they define the underlying unitary view of what constitutes the discipline, how it operates and its effectiveness.

The conception of a discipline consists of several parts:

- *discipline knowledge*, which can be formal, "...expressed in laws, theories and principles..." (Long and Dowell, 1989, p 11) or informal, "...only embodied in procedures and tools." (*ibid.*, p 11);
- *practice*, which can be either scientific (explanatory and predictive) or engineering (design) in character, devoted to solving the discipline's general problem;
- a *general discipline problem*, which the practitioners in the area seek to solve or address employing the knowledge in their practice;
- a defined and therefore limited *scope*, which sets out the border between a discipline and all other disciplines; and
- the ability to be divided into *sub-disciplines* by a process of decomposition.

It should be possible to characterise any discipline using these factors. This would allow them to be compared and contrasted, and the interactions and interdependencies between them to be made explicit.

2.4.1.2 Definitions of Science and Engineering

Long and Dowell (1989, p 12) consider the difference between science and engineering to be that between the natures of their general discipline problems.

Science's problem is to explain and predict; engineering's is that of design. Each discipline is formed of a group of sub-disciplines, differentiated by the scopes of their general discipline problems, such as the sciences of psychology and biology.

---

10 Long and Dowell introduced their categorisation of a discipline in the context of a critical examination of the current state of Human Computer Interaction ("HCI").

2.4.1.3 Defining the Conceptions

Using the above definitions, Long and Dowell (1989, p 11) have identified three conceptions of HCI:

- a *craft*, in which domain knowledge is implicit in the practice and all rules are heuristic, with advances in practice or product being made by trial and error;
- an *applied science*, in which the fruits of scientific investigations are applied to the solution of the discipline problem, but that application is controlled only by informal rules; and
- an *engineering discipline*, in which the rules and their application are made explicit and formal.

Initially, once a need for some artifact is identified, it will be made by trial and error, and this will over time result in the development of a set of *craft* skills. These may be extremely sophisticated, and result in artifacts which perform well in use, but the process is characterised by a lack of understanding among its practitioners of why they achieve their results, or of how to go about improving the product without a further iteration of the trial-and-error process.

There is no theory underlying such work. The process is repeatable, in that the workers can make more items like the existing ones with predictable timescales and for a predictable cost. However, they find it difficult to estimate, by other than craft experience and rough analogy, the timescales and costs of building something which they have not done before, since they recognise the variable and unpredictable nature of their main developmental process, namely the examination of full-scale prototypes. Even where the results of scientific work can be employed to assist in the design of new products, this is in terms of particular improvements resulting directly from individual experiments, without the assistance of any more general rules or theories.

In general, a craft discipline is characterised by Long and Dowell (1989) as the use of heuristics rather than of scientifically-determined rules to support the necessary practices, and as the development of knowledge via a series of trial and error cycles. The craft's practice is improved by a mechanism analogous to Darwinian

evolution – see Dasgupta (1991, p 78), for an equivalent view of design in computer science.

Long and Dowell explain that the effectiveness of a discipline based on craft practice (Long and Dowell, 1989, p 18) is limited, since the domain knowledge:

- is either informal or implicit in the process, and therefore cannot be applied directly to problems. Being heuristically based, a successful practitioner is necessary in order to apply it;
- is not testable, and users therefore cannot be sure that the desired effect of the practice will be achieved; and
- is not generalisable.

An *applied science* is characterised by the adoption of the results of scientific research in relevant disciplines as an aid in the process of designing and developing the product. In Long and Dowell's words, the discipline "...recruits scientific knowledge to the practice of solving its general problem." (Long and Dowell, 1989, p 20). This scientifically derived information is "...explicit and formal, operational, generalisable and testable..." (*ibid.*, p 20) within its original research domain, but, when it is applied by practitioners in another area to solve their general design problem, it cannot be prescriptive. Long and Dowell (1989, p 21) instance as an example of applied science practice the application of psychological knowledge to HCI practice.

Long and Dowell (1989, p 23) note the following problems which restrict the efficacy of an applied science in practice:

- the scientific knowledge brought in to support the practice is not prescriptive, and cannot therefore be applied directly;
- the selection of the science to be applied is itself based on heuristics rather than formal rules;
- the guidelines derived from scientific practice do not guarantee the performance of the product; and
- the cost of acquiring the scientific knowledge can be high, although the subsequent cost of developing heuristics for applying it is low.

Additionally, since the knowledge is applied in a non-prescriptive manner, there is no assurance of either quality in the product or predictability in the process.

When it becomes an *engineering discipline*, a discipline achieves predictability of process and product by the adoption of prescriptive knowledge forming a set of rules for specifying designs before they are implemented, in the knowledge that they will work as expected. Here, the designers can "...design for performance..." (Long and Dowell, 1989, p 24), optimising their designs in a way not open to those who try to improve their designs by trial and error. Additionally, a decomposition-based approach can be adopted for the design process, with the assurance that each level is a representation of the previous.

For a discipline to accord with this conception, its knowledge must be "...formulated as engineering principles..." (*ibid.*, p 24). Long and Dowell note in the same place that, in order to be able to conceive of a discipline as reaching this stage, the discipline knowledge must be capable of being set out in a "...codified, general and testable formulation...".

The discipline which has become an engineering discipline benefits from the predictability of solutions based on its prescriptive rules, allowing problems to be solved reliably in predictable timescales.

I suggest that the borders between the three conceptions of a discipline are indistinct. At any time, any process can be considered to be in one of them or in a transitional phase between two, since a discipline's position can be considered to be a spread based on the positions of all of the examples of the process which it subsumes. The change from one conception to the next can be slow, and will often be characterised by arguments among practitioners as to the correctness of each proposed step forward.

2.4.2 Where Is Software Engineering?

Given this taxonomy, can we determine at which stage Software Engineering currently is? The evidence which could be adduced for and against locating it at each stage is set out here in general terms.

2.4.2.1 Evidence Concerning Software Engineering as a Craft

Evidence which can be cited using the above definitions in considering Software Engineering as a craft include:

- the lack of understanding of, or rules which set out how the process works – see for example van Vliet (1993, p xviii), citing comments that Software Engineering cannot be taught, only preached, and promoting his view of the need for practical exercises to teach software development, perhaps reflecting the lack of anything more deeply theoretical to teach;
- the craft basis of development of new process models and methods, which seem to result from trial and error modification of existing methods with an admixture of new ideas, rather than from the application of theoretical principles; and
- the small number of applications of scientific principles to assist in the development of the software process or its products.

2.4.2.2 Evidence Concerning Software Engineering as an Applied Science

Evidence for Software Engineering currently being an applied science, to the extent that the results of scientific experimentation and theory are included in practice, include:

- the use of some scientific research work to support practice, such as the employment of:
  - psychology in HCI practice for developing and testing user interfaces (see, for example, Dix *et al.*, 1993);
  - program metrics in testing for quality (see, for example, Fenton, 1991); and
  - program metrics for performance;

  in developing the product; but there are fundamental disagreements over the meanings of how to define quality and performance and of the value of using currently available metrics, inhibiting the application of more scientific results;
- the application of work on the psychology of notations, programmer thought processes and so on (Miller, 1956; also Petre, 1995); and

- developments in styles of system development arising from work in programming machine models, such as object-oriented methods.

These applications of the science base to CBSD bear some resemblances to Long and Dowell's view expectations of an applied science discipline:

- the *ad hoc*, informal nature of the selection of the scientific principles – although the selection may be limited by the current state of the relevant science base, in this case 'computer science' – to be applied to Software Engineering practice; and
- the lack of prescriptive rules with predictable results derived for Software Engineering practice from scientific research.

Additionally, most of these applications of a science base relate to the design or implementation of computer-based systems; analysis seems still to be a craft.

2.4.2.3 Evidence Concerning Software Engineering as an Engineering Discipline

The following points could indicate that Software Engineering has obtained, or is in the process of obtaining, some of the necessary attributes to be considered an engineering discipline in Long and Dowell's terms:

- designs are specified before they are implemented, and they (mostly) work as expected; and
- the development of CBSD methods, and an understanding of their tailoring for specific applications.

Evidence against Software Engineering being a true engineering discipline as described above includes:

- the lack of a universally agreed conception or paradigm of the general design problem of Software Engineering;
- the lack of a universally agreed definition of how to describe the 'quality' of a finished software product[11] (as will be seen later in this thesis);

---

11 I ignore the contentious point of when a software product is 'complete' at this stage; my reticence may stand as another criticism of the current state of Software Engineering.

- the lack of prescriptive rules for the development of new computer-based systems in a reliable, predictable manner;
- the understanding of procedures and notations only in the context of the process, and not outside it (for example, notations developed as one view of 'information' may answer only those questions which a particular process requires);
- the absence of claims for predictability, or for pre-calculated metrics of *process*, in what are considered to be the best of new methods, such as for example, OMT (Rumbaugh *et al,* 1991); and
- the lack of well-defined and reliable tools to predict the effect of changes in the process on product quality or process resource consumption, when, for example, a different process model is introduced, or when a process has to be modified to meet particular circumstances.

I have been unable to find any references falsifying the above assertions.

2.4.3 Conclusion

From the above evidence, it can be seen that a case could be made out for Software Engineering being simultaneously at all three stages. This is undoubtedly characteristic of a discipline in a state of rapid development. However, the evidence in favour and against Software Engineering being at each level is variable in quality.

The weakest evidence is that for it being an *engineering* discipline, in that we do not yet have a prescriptive understanding of the product, and lack generally accepted theories to underly its performance. Software Engineering has undoubtedly progressed beyond the most primitive stages of the craft, but the application of scientific principles, to make it an *applied science*, is weak and patchy, and heuristics for the use of such principles are in the early stages of development.

In Long and Dowell's terms, the current state of Software Engineering is therefore mainly that of a *craft*, but with some of the attributes of an *applied science*, namely the pragmatic application of techniques and information derived from a science base, to be found in predefined techniques and methods. That the application of these rules can be somewhat haphazard, since the application of methods and techniques is still subject to judgement, occasionally supported by heuristics, as to

their relevance to any particular application, indicates that the goal of 'Software Engineering' explicit in its name, *viz.* the status of an engineering discipline, has not yet been achieved in Long and Dowell's terms.

2.5 A Critique of Computer-based Systems Development

If Software Engineering appears to be more of a craft than an engineering discipline, then what of the more general activity of CBSD?

2.5.1 In General

The current state of CBSD is that of a discipline fragmented into sub-disciplines. Some developers base their work on predefined methods, whether home-grown or imported, while others rely on *ad hoc* design mechanisms. There is no general agreement as to whether a predefined process model and/or set of tools is necessary in order to develop software effectively, and no agreement in specific areas as to whether the tools on offer are actually as effective as their proponents maintain. There is not even agreement concerning what comprises the discipline of method development, and how to study method.

To an outside observer, the current state of CBSD reveals a discipline divided into what might be termed religious groups, each espousing their own version of the Truth, whether it be formal methods or object-oriented modelling. The common thread is the lack of a universally-espoused underlying theory of what they are doing and how to approach doing it.

2.5.2 Some Explicit Failings

2.5.2.1 Failure to Meet Requirements and Deadlines

There is a perception among the general public that current CBSD mechanisms are unable to produce systems which meet the requirements of their purchasers and users. This perception is fed to some extent by a small number of dramatic, publicly reported failures of CBSD projects. A report on one of these failures, that of the London Ambulance Service (LAS, 1993), will be used later in this thesis as a source of information.

To some extent, this public view matches a perceived difficulty within CBSD of meeting a system's requirements within given deadlines.

2.5.2.2 Individual and Informal Developers

As an example of how predefined CBSD mechanisms are not universally perceived as meeting their declared objectives, it is useful to ask how those people who develop computer-based systems on their own and/or without the use of predefined methods see the influence of Software Engineering on their work.

From anecdotal evidence, and the semi-formal interviews with software practitioners described later in this thesis, it appears that those who develop computer-based systems without reference to a predefined method, often in small teams or as individuals, feel ill-served by the current range of formalised CBSD tools. They see these tools as mechanisms for managing and coordinating large development teams rather than for supporting the imaginative and communicative processes of CBSD. Such formalisms are therefore seen as being irrelevant to their work. They prefer to use informal analysis and design mechanisms and notations to the formalised mechanisms advocated by the authors of textbooks and courses.

The detailed comments can be summarised as being that current formalised CBSD mechanisms:

- do not meet the needs of the individual and/or informal developer;
- are more focussed on the needs of the development group building large systems;
- reflect concerns related to standardising notations and documentation, to enhance project control and reduce the reliance on specific individuals;
- are oriented towards project management; and
- represent in total an overhead on the individual developer which does not provide sufficient advantage to that individual to justify informing him- or herself about them and employing them.

Whilst the interview results arise from an informal investigation rather than a scientifically-based survey, the responses showed sufficient consistency to suggest that they are representative of at least a significant minority of informal or individual computer-based systems developers.

The implication of the above views is that the current tools offered by the theorists of CBSD do not provide sufficiently useful support for the developer working alone to persuade him or her to adopt them without pressures from above. The developer working in an informal group may also be put off by the implicit and explicit formalisation required on the adoption of a predefined method.[12]

All of this fits in well with the focus explicitly adopted by Sommerville, and demonstrated by his comments reproduced above on definitions of Software Engineering.

2.5.2.3 The Lack of an Underlying Theory

Computer-based systems developers seem often to know 'what' they are doing, but not 'why' it works, as might be expected from the practitioners of a Long and Dowell craft. This lack of fundamental understanding is demonstrated in the mechanisms used for teaching, which are based mostly on learning by doing, not from an underlying theory.

There is no *explanation* of the influences on systems developers of the tools they use. This is exemplified in the work of framework and method developers, who sometimes assume that one tool can be substituted for another without affecting the process; consider Avison and Wood-Harper's (1993, p 244) support for the substitution of one modelling tool for another in Multiview.

More generally, whilst I do not necessarily accept Episkopou and Wood-Harper's (1986, p 222) subdivision of CBSD approaches,[13] I agree with the general sentiments which they express when they comment in respect of systems analysis that:

> "Once the choice [of one or more approaches, or views of the world – General Systems Theory, Human Activity System, Socio-Technical/participative or structured systems analysis – a taxonomy which they use to divide CBSD analysis mechanisms into groups] has been made, ... the analyst has

---

12 One interview subject (Subject G, transcript, p 8) reported that a predefined method, in this case SSADM, had been greatly simplified and made less formal to make it more suitable, in the method developers' view, for small development groups working informally.

13 See Chapter 8 below for a discussion of this aspect of their work in the context of the work presented in this thesis.

put on his/her chosen 'glasses' with which to view the problem situation."

However, there is as yet no complete description of the effect of these 'glasses', nor of how that effect is obtained.

Evidence of a belief among some at least of the systems development research community, that the process of developing software may affect the product, is provided by such mechanisms and activities as the drive towards user-centred design, the Scandinavian School of Participatory Design,[14] and Wood-Harper and Fitzgerald's (1982) system vs. science paradigm dichotomy, but there is currently no effort towards locating and examining *all* of the beliefs underlying CBSD, either individually or in each other's context. The examples given of theory affecting practice each only considers a limited number of these beliefs, such as the place or importance of the user, or definitions of 'system' and its place in the world. The main thrust of this thesis is towards developing a theory and a framework for considering these beliefs as a group; its relationship with the work of others is also considered later.

It is interesting to note that some if not all presenters of methods, particularly in the textbooks from which new generations of practitioners are taught, do not mention, let alone emphasise, the belief-based aspects of the methods they teach, concentrating rather on the mechanistic aspects of systems development – see, for example, Sommerville (1992) and van Vliet (1993). This is noticeable, for instance, in the use of different computational models for modelling notations without consideration as to whether this affects the models obtained or the quality of the resultant real-world representations.

2.6 The Problem Statement

Based on the above critiques, I can now describe more definitely the problem which I am setting out to address:

> *The current state of Software Engineering does not address the concerns of those who develop software-based systems informally. This is a symptom of a wider malaise, namely the lack of complete understanding of the beliefs implicit in the*

---

14 "Cooperative design, as developed in Scandinavia over the last decade, stresses the importance of creative involvement of potential end users in design processes in general" (Groenbaek *et al.*, 1993, p 67)

mechanisms currently in use, and the effects of these beliefs on CBSD practice.

Can this problem be addressed using any existing intellectual tools?

In the next chapter, I will introduce the work of a discipline which provides a mechanism for addressing this problem; the Philosophy of Science.

2.7 Summary

In this chapter I have defined terms which have allowed me to describe the scope of the discipline under examination. I have also shown that the current effectiveness of that discipline is not as great as might be wished, with particular regard to those who develop their software in an informal manner.

I have also refined the problem addressed by this thesis, and suggested that the Philosophy of Science might help in further consideration of the issues.

3 Concerning the Philosophy of Science and Computer-based Systems Development

3.1 Introduction

In this chapter, I advance arguments for the use of theories from the Philosophy of Science as a basis for examining and improving the current state of CBSD. I then consider selected works of philosophers of science, and choose one theory to provide a basis for subsequent research.

3.2 Why Use the Philosophy of Science?

3.2.1 Introduction

In this section I present three views of how the use of the Philosophy of Science in explaining and supporting CBSD may be justified. This forms a precursor to a description of the philosophies of science themselves.

3.2.2 A Particular Argument

There is an observable trend among those who seek to divide CBSD into sub-species of identifying a 'science' element in its practice. Consider the work of:

- Hirschheim and Klein (1989), in which I identify the 'science' approach with their objectivist-order position, which they term the 'functionalist paradigm'; and
- Wood-Harper and Fitzgerald (1982), particularly their references to two different types of approach to software development, which are designated 'science' and 'systems', and used to classify a set of six lower-level approaches.

On this basis, if we are to examine how CBSD works, and how it might or should work, and the influences on its practice, we might look to a discipline which has been performing the same task for science. The rationale for the Philosophy of Science has been to explain and guide the work of scientists, and I wish to do similar work for computer-based systems developers.

3.2.3 A General Argument

The Philosophy of Science is a discipline which seeks to look at another discipline's practices with a view to understanding and improving the latter's theory and practice. I wish to do the same for CBSD.

I have therefore taken the Philosophy of Science as a starting point, and considered the possible adoption of those of its theories which have some apparent relevance to the current state and practice of CBSD.

3.2.4 Pragmatism

Perhaps the most easily defensible way of justifying my application of the Philosophy of Science to CBSD is an appeal to unadorned pragmatism.

As will be seen later, a model drawn by analogy from the Philosophy of Science works well in describing the current state of computer systems development, and provides new ways of approaching the perceived problems of the latter discipline.

3.3 Some Major Strands in the Philosophy of Science

The theoretical strands forming the Philosophy of Science are complicated and intertwined. The analysis presented here divides these into three levels of analysis of scientific activity, based on the elements of science examined by the philosophical theory. This division simplifies consideration of the possible uses of the Philosophy of Science in advancing the state of CBSD, which is summarised at each level using the examples given.

Some work in the Philosophy of Science relates to the nature of the theories by which scientists seek to explain and predict individual observations. Other work relates to the way in which scientists build larger theory bases, and yet other work to the way in which scientists work and relate to each other and to the rest of society. The three levels of science presented here are therefore those of the scientific theory, the theory structure and the science. One or more well-known examples of a theory from the Philosophy of Science is presented for each level.

The subdivision presented here is simplistic in that some of the theories of philosophies of science relate to more than one of these levels. However, the

taxonomy is still useful to make the division in order to see how these ideas can be used to support the theory and practice of CBSD.

3.3.1 The Specific Theory – e.g. Popper: 'Falsification'

The philosophical concepts at this level relate to mechanisms for building and selecting specific theories of science. Popper's concept of 'falsification' is a good example of this. Chalmers (1992) describes this concept as the idea that, since it is logically impossible to prove a theory to be true, scientists should bend their efforts towards testing theories in such a way as to demonstrate the falseness, regarding 'failure' in this endeavour as a reason for maintaining some current, temporary belief in the theory. Theories cannot be proven, therefore their status of acceptance can never be more than tentative.

More fundamentally, no theory can be regarded as scientific *unless* it can be subjected to a test that can result in its being falsified (Chalmers, 1992, p 41), and theories which can be easily falsified ('bold conjectures') are better than those which are more difficult to falsify.

Popper's ideas on falsifiability have gained some currency in the discipline of CBSD as principles for testing systems in a manner analogous to his view of testing scientific theories.

3.3.2 The Theory Structure – e.g. Lakatos, Feyerabend

Here the philosopher of science has concerned him- or herself with the complete body of theory of a science.

3.3.2.1 Lakatos: Research Programmes

Lakatos' framework for scientific progress, The 'Methodology of Scientific Research Programmes' (Chalmers, 1992, p 80), revolves around the concept of 'research programmes'. These are theoretical structures which can be used to guide scientific research. Lakatos' methodology is not an observation of existing practice, but a design for the future practice of science, and is thus very different in concept from Kuhn's observation-based, descriptive work. Lakatos' work arose as a response to Popper's ideas on falsification, which seemed to fail to explain the coherence of scientific thought and progress (*ibid.*, p 80).

A Lakatosian research programme consists of a general structure for theories, and a way of testing and extending these theories. The structuring of the theories divides them into a *hard core* of general, high-level assumptions basic to the research programme and which cannot be replaced within that programme,[15] and a *protective belt* of hypotheses, initial conditions, and hints and mechanisms for changing or improving the protective belt itself. The hard core is inviolate due to a methodological decision by the scientists in the research programme (*ibid.*, p 81); any problems of theory or observation are attributed to other parts of the theory, i.e. the protective belt, which can be modified if necessary.

Lakatos combines this theoretical structure with two heuristics. These are:

- the *positive heuristic*, which describes how the programme is to be developed, by adding assumptions to the protective belt and by predicting novel phenomena; and
- the *negative heuristic*, which is the underlying stipulation of the hard core.

If a scientist detaches his or her thinking from the hard core, he or she has then opted out of the research programme itself – belief in that hard core is a condition for membership of the research programme. That belief in the hard core is a matter of *belief*, rather than rational judgement (*ibid.*, p 81).

According to Lakatos, scientific research is a matter of expanding and improving the protective belt, perhaps by defining new theories or improving investigative mechanisms such as experimental equipment. Neither *ad hoc* changes to theories, nor modifications to the hard core, are allowed. Order in the theory and coherence in the research programme are maintained by the existence of the hard core.

Research programmes are classified by Lakatos according to whether or not they are leading to the prediction of novel phenomena; they are either *progressive* or *degenerating* (*ibid.*, p 80). Competing programmes are to be compared on this basis, and winners determined. However, even Lakatos himself observes that such victories can often only be seen with hindsight, and the problem with his

---

15 Chalmers (1992, pp 80-81) instances the hard core of Copernican astronomy, which is that the earth and planets orbit round a stationary sun, and that the earth rotates once every 24 hours.

methodology of apparently degenerating programmes being reinvigorated by new discoveries is difficult to overcome (*ibid.*, pp 86–7).

Research programmes could be paralleled in CBSD either with the emergence of families of development methods, such as the object-oriented model for design and programming, or with application areas. Here the hard core of object oriented thinking has been supplemented with the protective belt of specifics of object types and software design and development methods. The positive heuristic might be as simple as the extent to which a particular method or family of methods works, and how 'works' is defined in this context. ' The use of Lakatos' work in CBSD is suggested under 'future work' in the conclusion to this thesis.

3.3.2.2 Feyerabend: The Pragmatic Application of Investigative Methods

To Feyerabend (1975), the idea that a predefined method can be applied generally to solve the problems of science is untenable; whatever is best under the circumstances should be employed. The difference between the scientist and the crank is that the former will experiment to test his or her hypotheses, whereas the latter merely *believes*. However, science is not of itself necessarily superior to other intellectual fields (*ibid.*, p 205).

Turning to the question of how to test hypotheses, Feyerabend is the ultimate pragmatist, unfettered by preconceptions as to the process (*ibid.*, pp 295–296):

> "The idea that science can, and should, be run according to fixed and universal rules, is both unrealistic and pernicious. It is *unrealistic*, for it takes too simple a view of the talents of man and of the circumstances which encourage, or cause, their development. And it is *pernicious*, for the attempt to enforce the rules is bound to increase our professional qualifications at the expense of our humanity. ...
>
> All methodologies [of science] have their limitations, and the only 'rule' which survives is 'anything goes'".

If this idea were to be applied to CBSD, it would strike at the very concept currently at the heart of method-based systems development, that is of predefining a complete set of tools to be used in a particular order. In a restrained form, this could be regarded as favouring a more open mind on the part of a systems

developer in selecting method, tools and techniques. Ultimately, it would favour more pragmatic, unplanned process models based on *ad hoc* adoption of tools and techniques.

3.3.3 The Science – e.g. Kuhn: Scientific Communities and their Belief Systems

Kuhn's work considers both how scientists combine into communities, and how each of these communities is united by a common underlying agreement on principles.

In examining the basis of revolutions in scientific thought, Kuhn has introduced two key concepts. These are:

- the scientific community – the group of 25–100 scientists active in a scientific discipline; and
- the 'paradigm' or 'disciplinary matrix' ("DM"), which is a science's belief system, the concepts, models and beliefs which form the shared language of the practitioners in the science in a manner similar to Lakatos' hard core and protective belt. Kuhn's suggestion of a non-rational reason for accepting the paradigm is also reminiscent of Lakatos' account. This paradigm forms the unchanging basis of 'normal science' activities, but is modified from time to time as the result of 'scientific revolutions'.

Before it can become a science, a discipline goes through a stage in which it is divided into competing schools, each of which is united by a different paradigm. These schools argue over such matters as the definitions of terms and the meanings of observations, until one paradigm overcomes all opposition, and all of the schools combine under it. At this point, a Kuhnian 'science' has been established.

It can be observed that elements forming the DM of a science can be at either of the two levels already described, the individual theory or the theory structure. However, Kuhn adds general beliefs on how to build theories, values such as accuracy and support for numerical results, to provide a much richer view of a scientist's belief system than can be provided by either of the two lower levels.

On the basis of this outline description, it is apparent that an analogic connection to Kuhn's work can provide the basis of an examination of the underlying beliefs and models forming the discipline knowledge of CBSD. This will be expanded on later in this chapter, when Kuhn's work is described in more detail and explained in the context of CBSD.

3.4 Selecting a Theory – Initial Thoughts

Given the differing viewpoints of science taken by philosophers of science, it is necessary to select one strand to provide a theoretical base for the work which follows. Whilst a more detailed explanation of the rationale for my choice must follow a fuller description of the selected theory, I provide here an outline of why I have used Kuhn's view of science as the basis for my work.

I suggest that the current state of the discipline of CBSD looks somewhat like Kuhn's view of a discipline which has not yet become a science, particularly in its divided state and degree of mutual incomprehension, and that the development of a computer system using a predefined method may bear some resemblance to the Kuhnian view of normal scientific activities, working within a generally unquestioned belief system. These similarities are sufficient to justify the more detailed look at Kuhn's work which follows.

The term 'paradigm shift', which describes a change in DM which results from a scientific revolution, has been much used (and abused) in the literature of CBSD and other fields. Indeed, it has now become no more than a marketing term in many contexts, although it still appears in the CBSD literature (see, for example, Connaughton and Dampney, 1994). Its use begs the question of the nature of the paradigm which has been shifted. I feel that this in itself is an interesting enough question to justify further investigation of Kuhn's theories.

The analogy between Kuhn's view of science and the current state of CBSD is at this stage purely conjectural, being based on a superficial similarity between the two. To this extent, the selection of the work of Kuhn, rather than that of another philosopher of science, is arbitrary. Until the analogy between Kuhn's work and CBSD has been examined in detail, there is no evidence supporting the selection of his work before that of others.

In the next chapter, having described Kuhn's work in more detail, I will explore the analogy and form a model of CBSD theory and practice firmly based on Kuhn's theory. This is intended to demonstrate the use of Kuhn's work in building a theory for CBSD. In Chapter 5, the analogy between Kuhn's work and the current state of CBSD will be tested informally in a number of investigations. Together, these provide evidence of the reasonableness and power of the analogy which I am making.

3.5 The Work of Kuhn in More Detail

3.5.1 Introduction

It is necessary for me to provide more detail on what Kuhn's theory is, before its application to describe CBSD can be considered. The following description of Kuhn's work is based on the second edition of The Structure of Scientific Revolutions (Kuhn, 1970) and on Second Thoughts on Paradigms (Kuhn, 1977), as well as the outline of Kuhn's work set out in Chalmers (1992). The description is not intended to serve as a general introduction to Kuhn's work, and covers the ground in a form most suitable for my subsequent application to CBSD.

3.5.2 Two Important Concepts

I describe here two of the fundamental concepts in the Kuhnian view of a science; the scientific community, and the paradigm, or DM, which unites and underlies the work of a scientific community.

The *Scientific Community* is a separately recognisable group of people, comprising the practitioners in a scientific speciality (Kuhn, 1970, p 177). The group tends to be small (25–100 people), and is characterised by the unified thinking of its members. "To an extent unparalleled in most other fields, they have undergone similar educations and professional initiations; in the process they have absorbed the same technical literature and drawn many of the same lessons from it." (*ibid.*, p 177).

Each of these communities can be seen as one of the leaf nodes of a tree of disciplines and sub-disciplines, from the most general discipline of 'natural science' to the levels of the communities.

One of the characteristics of a scientific community is the high level of agreement among its members on general metaphysical principles. Any disagreements, dividing the community into schools based on these differences, are more quickly resolved than in other types of discipline, due to the open communication among the community members allowed by their shared assumptions (*ibid.*, p 177). These assumptions form a set of metaphysical beliefs, which cannot be verified from within the discipline which they define. Kuhn calls this set of assumptions a 'paradigm' or 'disciplinary matrix' ("DM").[16] The existence of this paradigm in the form described here differentiates a science from a non-science (Chalmers, 1992, p 91; he gives an example of a non-science as "...much of modern sociology...").

The effect of the DM on a scientific discipline is to define what can and cannot be done within it. The DM sets the problems which can be solved, identifies which are the most urgent, and indicates what constitutes a valid solution to a problem.

Although the DM is agreed upon by all members of the scientific community, albeit that some elements of the DM might be implicit rather than explicit, different interpretations of the beliefs enshrined in the matrix may result in different decisions or strategies; as Chalmers (1992, p 99) puts it, science hedges its bets.

It is in the nature of a DM that it cannot be defined precisely. Types of elements can be described, such as fundamental laws and standard ways of applying them (Chalmers, 1992, p 93). However, the elements of the DM can be defined sufficiently well to defend it when crisis approaches (Chalmers, 1992, p 93).

Despite this imprecision, Kuhn describes some of the types of elements which he expects to see in a DM. These are outlined here.

Symbolic Generalisations

These are "...those expressions, deployed without question or dissent by group members, which can readily be cast in a logical form..." (Kuhn, 1970, p 182). If not cast in explicitly symbolic form (such as mathematical formulae), they are

---

16 The latter term was introduced in the postscript appended to the second edition of 'The Structure of Scientific Revolutions', in order to limit the confusion caused by the use for a variety of meanings of the word 'paradigm' in the main text. I have used the term 'disciplinary matrix' here, usually abbreviated to "DM", to describe the belief system, but the tendency amongst philosophers of science seems to be to use the term 'paradigm'.

expressed in words; Kuhn adduces the example of 'action equals reaction' (Kuhn, 1970, p 183).

The importance Kuhn ascribes to symbolic generalisations is, in part, that of scientific laws, but additionally that of defining the terms used in those laws; the balance between law and definition changes over time as understanding grows in the community.

Belief in Heuristic and Metaphysical Models

Kuhn refers to metaphysical models in the main body of his monograph by such descriptions as "...'metaphysical paradigms' or 'the metaphysical parts of paradigms'..." (Kuhn, 1970, p 184). They are "...shared commitments to such beliefs as: heat is the kinetic energy of the constituent parts of bodies..." (*ibid.*, p 184) and so on. They "...supply the group with preferred or permissible analogies and metaphors." (*ibid.*, p 184), defining what is an acceptable explanation or solution in the eyes of the community. Kuhn also includes in this group heuristic models used to assist in the formulation and testing of scientific laws; the billiard-ball atom model of a gas can be included in this category.

Many of the instances which I have found of the use of the term 'paradigm' in the CBSD literature refer to models or beliefs which can be included in this category, either for modelling the real world or software products (such as the 'object-oriented paradigm'), or as heuristic models of the software development process.

Values

These are "...more widely shared among different communities than symbolic generalisations or models." (Kuhn, 1970, p 184). They are deep and general in nature, covering for instance the requirement among scientists for predictions to be quantitative rather than qualitative if possible, and the desire for simplicity and consistency in theories.

Values are often shared by different communities, but their application, for instance the exact value ascribed to the desirable objective 'accuracy', might vary from one discipline to another.

Exemplars

The exemplar is the "...paradigm as shared example." (Kuhn, 1970, p 187). Exemplars are examples of the application of the theories of the discipline to observations, whether as examples in textbooks or as the bases of great discoveries which have illuminated the area for future practitioners, taken as models by all practitioners in the discipline.

The science student studies these shared examples to gain knowledge as to how the theories which he or she is learning are applied to nature; "In the absence of such exemplars, the laws and theories he has previously learned would have little empirical content." (*ibid.*, p 188). Other exemplary paradigms published in technical literature "...show ... [practitioners] by example how their job is to be done..." (*ibid.*, p 187). This second sub-type of the 'paradigm as example' also helps to delineate the border between different disciplines, whereby the community of practitioners in one form of science accept a description or explanation of an example or phenomenon different from that believed by another.

3.5.3 The Incommensurability of Disciplinary Matrices

Despite the fact that Kuhn's theory of scientific change requires the comparison of DMs, as we shall see below, it is impossible for two DMs to be measured directly against one another. When it is necessary to select between DMs, this is therefore not a question of comparing the two; Kuhn states that this simply cannot be done. The DMs embody different sets of assumptions, and they use different terms, or different definitions of the same terms, to describe the world. Judged by its own values, one DM may be superior to another. However, the same may also be true in reverse, as the application of a different set of values may result in a different conclusion.

How then can a scientist decide rationally between DMs? Kuhn notes that "...proponents of competing paradigms must fail to make complete contact with each other's viewpoints." (Kuhn, 1970, p 148). He describes several reasons why this is so (*ibid.*, p 148 *et seq.*):

- disagreements over the problems to be attacked by the competing paradigms, due to differing definitions of 'science';
- the same vocabulary and apparatus are used in different contexts, and the results mean different things; and

- "...[the] most fundamental aspect ... the proponents of competing paradigms practice their trade in different worlds" (*ibid.*, p 150); they look at the same phenomena from such different viewpoints that "...they see different things, and see them in different relations one to the other." (*ibid.*, p 150)

For these reasons, Kuhn believes that adherents to different paradigms find it so difficult to communicate effectively that a complete understanding between them of the differences between their paradigms is not possible. Because DMs are incommensurable, the process of comparing one with another becomes one of *translation* (Kuhn, 1970, p 202).

It is worth highlighting the point that the differences in paradigm which result in incommensurability are in part those of *translating* one set of ideas into a different vocabulary, or the use of the same vocabulary for different purposes.

3.5.4 Kuhn's Characterisations of The Life Cycle of a Scientific Discipline

The life cycle of a scientific discipline, according to Kuhn, can be described as consisting of:

- a *pre-science* stage, with different schools of thought coalescing about competing DMs, each defining and redefining the metaphysical principles of the discipline, ending with
- coalescence into a *science* when one DM achieves pre-eminence over the others, and all practitioners agree on it.

The lifetime of this science is itself composed of:

- periods of *normal science*, pursuing the goals set with the means provided by the DM, punctuated by
- *crises* in the scientists' belief in the DM, which result in *revolutions* modifying the DM.

I now consider the characteristics of Kuhnian pre-science disciplines, sciences and non-sciences.

The Pre-Science Discipline

This comprises disciplines which are capable of developing into sciences, but have not got there yet. Kuhn gives the example of optics in the 17th Century and earlier, before the emergence of the first universally-accepted model of light (Kuhn, 1970, p 12 *et seq.*).

A pre-paradigm discipline is recognisable by the existence of many competing paradigms, each of which is identified with one of many *schools* (Kuhn, 1970, p 16) within the discipline. Each school thus has its own paradigmatic structure of beliefs and models (Kuhn, 1977), shared by its own school members but disagreed with to a greater or lesser extent by members of other schools. These schools:

- disagree over the nature of the phenomena with which they are dealing and how to interpret their observations (*ibid.*, p 17); in Kuhn's example of optics, light was seen by different groups as particles emanating from material bodies, modification of the medium between the body and the eye, or interaction of the medium with an emanation from the eyes (*ibid.*, p 12);
- are unable to agree on definitions of key concepts, and therefore talk at cross purposes (*ibid.*, p 198 *et seq.*); key concepts which have been the subjects of such disputes include 'element', and the distinction between chemical 'compound' and 'physical mixture', without which the Law of Definite Proportions is not able to be derived; and
- each have their own DMs (Kuhn, 1977), including different metaphysical bases and the exemplars which each of their theories do most to explain (Kuhn, 1970, p 12–13)

An important consequence of the lack of a shared belief system across the different schools of thought common among those examining a class of phenomena is the *incommensurability* of the differing paradigms. They see the same phenomena in different ways, using different terms (or, worse, the same terms with different meanings) to explain them. This results in the need to translate between the work of different practitioners, and a critical loss of synergy between the different groups.

It is interesting to note that Kuhn accepts 'the more creative' members of these groups as 'scientists' (Kuhn, 1970, p 13), even if a unified 'science' has not yet emerged!

Kuhn gives examples of other disciplines which have exhibited this early stage during their history – including chemistry, heat and motion – before becoming 'sciences' with a universally accepted paradigm; he notes (Kuhn, 1962) that other disciplines remained to have their 'paradigm' universally accepted.

The Scientific Discipline

At some point in the life of a pre-science, a single DM emerges, replacing all of its competitors by reason of its acknowledged superiority to them. This DM is accepted by all practitioners in the discipline. At this stage, to Kuhn the discipline has become a *science*. "A mature science is governed by a single paradigm" (Chalmers, 1992, p 90).

In contrast to a fragmented, school-based pre-science, a science is thus based on a single DM, which has emerged as the winner from the DMs of a set of pre-science schools, and which is now for most purposes agreed to by all practitioners. However, Kuhn also notes the "relative scarcity" of competing schools in the "developed sciences" (Kuhn, 1970, p 209) indicating that some fragmentation into schools still occurs and that some current 'sciences' have not yet completely achieved their own unifying paradigms.

In a science, the nature and effect of its DM is changed from those of pre-sciences. The DM of a science presents challenging puzzles, supplies clues to solutions and guarantees the clever practitioner success, which those of the pre-science schools did not (Kuhn, 1970, p 179). The paradigm gives a firm metaphysical basis for work, and a common language. It defines which questions can reasonably be asked, and which do not make sense. For example, looking for definite proportions for components of chemical compounds did not make sense whilst the difference between 'solution' and 'compound', i.e. physical mixing and chemical reaction, were not accepted. Whilst they were seen as the same thing, any combinations of mixable components seemed to 'work' until some saturation level was reached. The view of what was happening, even though it could not be seen, changed when the crucial difference between 'mixture' and 'compound' was accepted.

The most common activity in a science, *normal science* work, consists of research directed to solving problems within the limits set by the science's DM. It is this shared set of beliefs and commitments which, as stated earlier, sets the problems

of the discipline, the terms in which these may be approached to give a valid solution and the means of identifying what constitutes a valid solution. Kuhn describes normal science as puzzle solving within the paradigmatic structure, which can result in many advances but which is limited in scope by the paradigm. A major advance in the change from pre-science to science is that a pre-science's fact-gathering is replaced by more purposeful experimental activity in the science, because the paradigm sets the problems and acceptable solutions.

Normal science can also be described as "...detailed attempts to articulate a paradigm with the aim of improving the match between it and nature." (Chalmers, 1992, p 91) The paradigm underlies this work, and is set out in, and reinforced by, the writing and use of textbooks which are based on the assumptions in the DM, and contain exemplars in the form of explanatory features and test exercises.

To Kuhn, the paradigm within which normal scientific work proceeds arises as a nexus of the sharing of beliefs, models and so on among those people active in the field in question. This group is the *scientific community*, and Kuhn believes that it is usually between 25 and 100 strong. Larger groups may exist however; the particle physics community, for example, numbers some 2,000–4,000. However, this can be subdivided into smaller groups, such as those who research into weak interaction; this group probably comprises no more than 400 active workers.[17]

The life of a science is not a smooth path of advancing theory and practice; normal science is not the only activity. Work is characterised by phases of normal science punctuated by scientific revolutions, which are the only mechanisms by which the DM can be changed. Scientific revolutions are described in more detail in the next section.

Non-Science Disciplines

These constitute all disciplines which do not conform to Kuhn's account of a science (Chalmers, 1982, pp 108–109, quoting Kuhn, 1970, p 22), i.e. those disciplines which are not capable of being united under a single paradigm.

17 Russel Winder, personal communication.

3.5.5 Crises and Scientific Revolutions

Normal science is the devising, performing and analysing of the results of experiments within an accepted DM. However, over time, problems will emerge with the theories enshrined in the DM. These problems may relate, say, to theoretical problems which the DM cannot explain, or to observations which the DM has predicted incorrectly. These problems sometimes become so great that they lead eventually to a sense of scandal and a feeling in the community of scientists that 'something must be done'.

As the scandal of the current DM's insufficiencies grows, and in an effort to explain the anomalies, competing paradigms emerge. At some stage, one of these – or the original DM restated – emerges as the new, generally accepted DM; this change or restatement of DM is a *scientific revolution.* After the revolution, the new paradigm becomes the basis for another period of normal science. Textbooks are rewritten from the new perspective explicit and implicit in the new DM, and the revolution becomes invisible. Normal science can now continue until the next crisis emerges and another revolutionary period begins.

For Kuhn, normal science and scientific revolutions have different purposes. That of normal science is to develop the theories of a discipline within the context of an agreed set of principles. No effort need be wasted in reinventing these principles or arguing over them. Confident in their paradigmatic framework, scientists can develop their ideas, and plan and execute experiments to test them.

The need for scientific revolutions is that, in common with other models, any framework for scientific development is not perfect. Therefore, a discipline which failed to break out of an unsatisfactory DM when the implicit or explicit simplifications in that DM became unsupportable would be unable to meet all of its challenges. There is a requirement to be able to question and change the underlying bases of a discipline *when the need arises.* Scientific revolutions perform this function.

A complete scientific revolution can be summarised as:

- the onset of crisis;
- the emergence of competing DMs which are believed to resolve the scandal which caused the crisis;

- the selection by the community of a new DM from the candidates;
- the completion of the scientific revolution, as the last adherents of the old DM die out; and
- the rendering invisible of the revolution by a retrospective 'fixing up' of the theory base to match the new belief system.

Normal science works within the DM, which sets problems and defines acceptable solutions. Most failures in this process will be regarded as failures on the part of the scientist rather than the DM – Chalmers (1992, p 94) parallels the carpenter blaming his tools. However, some problems may remain unsolved, and any of these may bring about a crisis if it strikes at the metaphysical heart of the discipline's beliefs. This state of affairs may not of itself provoke the crisis necessary for a scientific revolution. Anomalies can subsist in a generally accepted DM. Other conditions may be necessary for a crisis to occur, such as a pressing social need to solve a problem or a large number of such anomalies.

Eventually, the workers in a discipline enter a period of "...profound professional uncertainty..." (Kuhn, 1970, pp 67–68; quoted by Chalmers, 1992, p 94). The crisis deepens when workers begin to lose faith in the current paradigm, and when a rival paradigm emerges the stage is set for a change in fundamental beliefs (Kuhn, 1970, p 91; in Chalmers 1992, p 95). The rival paradigms will differ in terms of the fundamentals of the universe they are dealing with; they will see the world as being made up of different types of things.

The crisis attending the failure of the current DM to answer questions thus finally results in a change of DM, or a paradigm shift; a *discontinuity* (Chalmers, 1992, p 90) in the development of the science. Although crisis is the usual precursor of paradigm change, it should be noted that Kuhn does not regard a crisis as an essential prerequisite for a revolution (Kuhn, 1970, p 181); the possibility of a quiet revolution is allowed.

The process of this revolution, resulting in the overthrow of an existing paradigm and its replacement by a new one, is one which cannot be reduced to a logical choice, for the following reason. How might a scientist arrive at a preference for one DM over another, especially in view of the fact that they cannot be compared directly? Kuhn mentions some criteria which might be employed to choose between DMs, such as "...accuracy of prediction, particularly of quantitative

prediction; the balance between everyday and esoteric subject matter; and the number of different problems solved...", as well as less important matters such as "...simplicity, scope and compatibility with other specialities..." (Kuhn, 1970, p 154; quotation taken from Chalmers, 1992, p 107).

However, he concludes that the specification of these values for selection is in the final analysis "...psychological or sociological..." (Kuhn, 1970, p 154; quoted by Chalmers, 1992, p 107). The change is a leap of faith, a gestalt switch or religious conversion (Chalmers, 1992, p 96), based on propaganda from the adherents of the new DM or on personal reasons such as an attraction in the problems set by the new paradigm or the values implicit in it.

A scientific revolution ends when the new DM is accepted by the discipline, perhaps only when the last adherents of the old DM, literally, die out. Textbooks are rewritten to reflect the new DM, and the old DM becomes a matter of purely historical interest.

Kuhn notes that it is difficult to see scientific revolutions retrospectively. This is because textbooks are rewritten to conform to the new DM in the post-revolutionary world, often explaining the history of the science until the date of writing as a continuous march of progress, noting each advance *in terms of the currently-accepted DM, not of the earlier DMs underlying that earlier work.* These textbooks therefore disguise the role and nature of previous scientific revolutions (Kuhn, 1970, p 137), and the revolutions themselves disappear in a seamless pseudo-history of the discipline which in fact never existed. Earlier workers are assumed to have been working on the problems and with the attitudes set out in the current DM.

3.6 Criticisms of Kuhn

Kuhn's ideas have not been accepted without argument. I therefore describe some criticisms which have been made of his work. I will comment in the conclusion to this thesis on the relevance of these criticisms to the work described here.

The criticisms have been selected, mainly from Kneller (1978) and Musgrave (1971), for their relevance to Kuhn's theories as set out in the second edition of

'The Structure of Scientific Revolutions',[18] rather than the ease of answering them with relevance to CBSD:

- the DM is not useful as a research tool, because the contents are not specified (Kneller, 1978, p 61);
- scientific disciplines advance by other than crisis and revolution, e.g. by resolution of competing theories (Kneller, 1978, p 61);
- the activities described by Kuhn as revolutionary (criticisms of theories, work on alternative theories, debates on fundamentals) are going on all the time, only reaching a peak in revolutions (Kneller, 1978, pp 61–62);
- normal science is not rational, because the paradigm is inviolate (Kneller, 1978, p 62);
- Kuhn provides no (rational) basis for choosing between theories (Kneller, 1978, p 63). Shapire (1964, p 35–36) asks how one can 'compare' incommensurable paradigms to decide when to take up the new one? This is relativism! For Chalmers (1992, pp 107–109) also, the lack of absolutes in Kuhn's paradigm selection mechanism in revolutionary phases is indicative of Kuhn's relativist position rather than the rationalist position to which scientists prefer to see themselves adhering.

  Kuhn (1970, p 199) himself seems to adopt a relativist position when he describes some of the values which might affect paradigm choice, such as accuracy, simplicity and fruitfulness, but notes that since these are values rather than absolutes, they can be applied differently by different people who agree with them, to result in different conclusions. However, Chalmers (1992, p 107) notes Kuhn's disagreement with any identification of him as a relativist;
- It is difficult to determine the difference between the articulation of different paradigms and different articulations of the same paradigm (Shapire, 1964, p 32);
- Kuhn's views lead him to deny the existence of progress in some scientific changes, such as Einsteinian over Newtonian dynamics, "...in a more fundamental sense than can consistently be extracted from his conceptual apparatus." (Shapire, 1964, p 37). Musgrave suggests that what Kuhn

---

18 Some of the criticisms set out by Shapire are relevant only to the first edition of Kuhn's work (1962), and have been answered at least to some extent in the Postscript appended to the second edition (1970).

doubts is the progress of successive paradigms towards the truth rather than their undoubted greater success in solving the problems of nature (Musgrave 1971, p 49);

- Kuhn advocates identifying the scientific community first, then deriving the paradigm from observations of it (Musgrave, 1971, p 39), although Musgrave (*ibid.*, p 39) suggests that "...most philosophers of science..." would regard the identification of the paradigm as the first priority followed by the identification of those scientists who adhere to it. Musgrave also notes the difficulties in identifying the community without reference to the paradigm (*ibid.*, pp 39–41). He notes that Kuhn suggested (Kuhn, 1970, p 176) that work was being or to be performed to do so, and refers to the use of reference lists, such as those available from the Science Citation Index, in an attempt to delineate the scientific community in an objective manner by seeing who cites whom. Musgrave rejects this approach on the basis that, without looking at the subject matter of what is referenced, no conclusions can be drawn as to the existence or nature of any agreement between the parties;
- Musgrave (1971, pp 42–43) notes that, despite the comparative difficulty of cross-community communication (Kuhn, 1970, p 177), not all scientists agree on the fundamental metaphysical beliefs within which they work even when they are performing normal science (Kuhn, 1970, p 180), and therefore asks what the paradigm can consist of, and whether normal science as defined by Kuhn actually exists. Kuhn (1970, p 180) in fact suggests that complete unanimity on all aspects is not essential in a community, but that consensus on some of the fundamental points is;
- Not all elements of belief shared between members of a scientific community are necessarily relevant to that community's scientific work, and they may disagree on some of the elements relevant to their work (see above) (Musgrave, 1971, pp 44–45). Musgrave (1971, p 44) says that "Clearly, we shall have to employ a philosophical thesis to demarcate between what is relevant to scientific work and what not."; and
- Musgrave dislikes Kuhn's use of the term "puzzle-solving" to describe normal science within the DM, pointing out that Kuhn's analogy of the crossword puzzle is invalid since the crossword, like the textbook problem which Kuhn (1970, p 189) also mentions, has a known solution whereas

scientific research does not (Musgrave, 1971, p 46 – he prefers the use of the term "problem-solving").

Despite these criticisms of Kuhn's work, I believe that his ideas are of use in explaining and improving the current position of CBSD. The criticisms will be reconsidered in the concluding chapter of this thesis, in the context of the work described in it.

3.7 An Example of the Disciplinary Matrix – Some Elements of My Own DM

An interesting example of the DM to me is that which has underlain the work presented here. I therefore describe here those elements of my own DM which I have been able to identify as being the basis of the research reported in this thesis. This set of DM elements is based completely on a conscious introspection which has proceeded through the process of researching for and writing this thesis.

In addition to forming an example of a DM, my DM as set out below also acts as a qualifier to the work presented in this thesis, in that I have in this work acted as my own observer and the results which I have obtained are inevitably subjectively based. This subjective filtering provides a second reason for including my own DM in this thesis, to document some aspects of that filter. Whilst acknowledging the existence of these elements in my own belief system, I have endeavoured during the work summarised in this thesis to maintain an objective stance towards the current theory and practice of CBSD.

3.7.1 Symbolic Generalisations

As yet I have found no symbolic generalisations in my approach to the work in this thesis. I believe that this reflects the broad qualitative theoretical sweep of the work presented, in contrast with the quantitative basis of Kuhn's work in physics, from which many of his examples of symbolic generalisations are drawn.

3.7.2 Belief in Heuristic and Metaphysical Beliefs and Models

The beliefs and models which I have employed in the work for this thesis include the following:

- there *are* some fundamental truths somewhere in CBSD theory and practice, either already described or waiting to be found;

- when complete, the DM will help us to explain many of the design decisions by Software Engineering theorists (methods, tools, techniques, notations, ...) and practitioners (model building in systems analysis and design);
- the 'scientific method' : theory –> experiment/investigation –> theory ... ;
- the three-stage model of a discipline set out by Dowell and Long; their three conceptions of a discipline used above to support my work; craft, applied science, engineering discipline;
- Kuhn's conception of the nature of a scientific discipline, used as a parallel in building a metaphysical model of the design discipline of CBSD;
- decomposition; as exemplified in the hierarchically-organised set of beliefs presented as my version of the DM of CBSD in Appendix 2 to this thesis;
- a General Systems Theory-like approach to the construction of belief systems, with sub-levels adding more detail corresponding either to the greater degree of specialisation or to the adoption of optional beliefs, of adherents of sub-disciplines. This will be most apparent in Chapter 7 of this thesis, in which an algorithmic mechanism of how the DM of a CBSD activity might be conceptualised is described;
- papers in the published literature are generally correct as to reports of work done and results obtained, but not necessarily so in respect of their authors' interpretations and opinions;
- a certain lack of respect for the current 'received wisdom' of Software Engineering, such as the belief in some quarters that the waterfall model is obsolete. I hope that this belief results in an attitude which is constructively critical;
- a belief that there may be some truths outside the received wisdom of theoretical Software Engineering as currently taught;
- a desire that my research should be of benefit to software developers in the real world, rather than being of purely theoretical interest. I have tried to make the results of, and advice arising from, this work available in a practice-oriented environment; see, for example, the latter sections of Winder and Wernick (1993);
- my feeling that what appears to be missing from a DM can be as instructive as what is present in seeking to determine the thinking behind the

phenomena observed, e.g. interview subject G's not mentioning portability as a criterion for quality in a software product;

- a belief that people generally do things *for a reason* when developing software, even if the reason is not
  - rational, e.g. fashion, and/or
  - directly known to them, i.e. it is implicit rather than explicit, and they cannot articulate it – consider Kuhn's comment that it may not be possible to determine the contents of the DM;

  we can therefore guess at why they are doing something by looking at what they do and considering possible reasons as to why they might do it.

  In addition, if people do things for a non-rational reason (e.g. fashion), then the process of CBSD can be improved by examining the effects of these reasons, making the thinking behind them explicit, and possibly replacing the activities with processes based on rational principles;
- a belief in the need to check theories by their application to the real world, or by looking for examples of their predictions in the real world;
- a belief in the need to change Software Engineering to give support to those not yet explicitly included in the Software Engineering community, especially individual workers – contrast Sommerville's definition of Software Engineering quoted in Chapter 2 above;
- a belief that the value of a theory lies in its ability to predict novel and unexpected results; it is not enough to state the obvious, and it has been one of my concerns that the present work should do more than this;
- a belief that the portability of software is compromised by any form of 'lock-in', in hardware, software (such as proprietary languages) or computer-based support tools for design and documentation;
- a belief in the validity of the investigative mechanisms which I have employed for trawling for elements of the DM of CBSD; and
- a belief that the number of words written in a session is a measure of progress; this might reflect a deeper belief that if there is currently no quantitative measure of quality in a product under development (in this case, the work towards this thesis), then any quantitative measure is better than none.

3.7.3 Values

I have evaluated my work against the following set of values in order to see whether they are useful:

- *objectivity* is a good ideal to which to work in academic research. This is reflected in an endeavour not to judge the elements of the DM of CBSD, and not to consider whether they might be 'good' or 'bad', or whether their effects are 'good' or 'bad', but only to identify them and place them in the structure.

  This desire for objectivity may arise from any or all of:

  - my early hard science background;
  - my own and others' experience of CBSD work;
  - my experience in developing small computer-based systems successfully without the aid of the predefined methods which textbooks seem to imply are necessary; and
  - discussions with friends who develop good systems of some size without methods,

  and it is operationalised in my determination when drawing conclusions from this work not to devise or espouse any particular CBSD method, tool or technique, in order to avoid becoming committed to that mechanism and thereby losing independence of viewpoint;

- my *perception of myself* when developing software as:

  - 'champion of the user', wanting a usable, working and aesthetically pleasing solution, but not committed to any existing user-oriented CBSD method; and
  - a 'hacker', due to a number of years' working in CBSD, most of which was as an individual developer rather than as a member of a large, formalised team;

  and

- a general feeling that manual systems are more adaptable to changing circumstances than computerised systems. This might, for example, result in a certain cynicism about CASE tools, especially as described by their adherents.

3.7.4 Exemplars

I can identify two examples of good research practice which have influenced the work presented here. These are:

- Petre's thesis (1989), for mechanisms such as investigation by recorded interview; and
- examples of action research such as Checkland (1990), and Avison and Wood-Harper (1993).

Another less directly relevant exemplar is my definition of 'professionalism' implicit in the description of the DM of CBSD, which may be coloured by my perception of Chartered Accountancy as an exemplar of a profession.

3.8 Summary

Pragmatically, Kuhn's explanations look like a good fit for what we have observed in CBSD. The current state of CBSD method development shows many of the attributes of a set of pre-science schools, such as the viewing of the same phenomena from differing viewpoints necessitated by the adoption of different computational models to model those phenomena. The 'puzzle-solving within a philosophical framework' nature of Kuhnian normal science is a good analogy for CBSD under the control of a predefined method.

This parallel will be developed in the next chapter of this thesis, and tested and extended in succeeding chapters.

4 Applying Kuhn's Work To Computer-based Systems Development

4.1 Introduction

In this chapter I intend to develop the ideas outlined in the preceding chapter concerning the significance of the Philosophy of Science for CBSD, by explaining the current position of CBSD in terms of the general outline of Kuhn's theories, referring to such aspects as:

- identifiable communities reflecting differing schools of thought;
- the need for translation between the DMs underlying these schools of thought resulting in people talking at cross purposes; and
- the incommensurability of some of the theories advanced.

This is followed by a detailed consideration of the relevance of some of the details of Kuhn's model to CBSD, and three caveats concerning the application of Kuhn's work to CBSD.

In Chapters 5 and 6, I will build on this by proposing a structure for the DM of CBSD after some investigations into the beliefs, values, models and so on which are the individual elements of that DM.

4.2 The Analogy Applied

4.2.1 Introduction and Discussion

I describe here, in detail, the way in which the current state of CBSD can be described in terms of Kuhn's model of the development of a scientific discipline. To this end I extend the argument set out in the preceding chapter as a basis for applying the model to CBSD.

Specifically, I consider the way in which Kuhn's ideas fit into the current state of that part of CBSD which is based on predefined methods, tools and techniques, examining both the development and the use of these mechanisms. I also consider in outline the applicability of Kuhn's ideas to those computer-based systems developers who work without the benefit of such mechanisms.

One problem dominates all others in applying the Kuhn-based analogy to research into CBSD methods. Kuhn's model of a scientific discipline provides a view of a group of practitioners unified by a shared common belief system which supports all of their work. Such a commonality of thought is conspicuous by its absence in the group of people developing CBSD methods.

One solution to this problem is to suggest that the part of CBSD devoted to developing methods and tools is in an analogous state to that of a Kuhnian pre-science discipline, supporting different schools as in Kuhnian pre-paradigm science. This situation can be observed, for instance, in the adherence of method developers to different *computational models* underlying the notations used to build and/or document world or system models. The mechanisms by which a particular computational model is adopted seem, in Long and Dowell's terms, to be more characteristic of a craft than an applied science or engineering discipline. It also provides a parallel for CBSD to that component of Long and Dowell's (1989) conception of a discipline, described previously, that requires a discipline to be divisible into sub-disciplines by a process of decomposition.

When the view turns to the practice of CBSD, developing computer-based systems, a different model is needed. As we have seen, Kuhn views the mechanism of normal science as being the development and exercising of theories relying on an unchallenged metaphysical base in which confidence is held. For Kuhn, it is necessary that the metaphysical base be assumed to be solid in order to perform the detailed work needed to explore its possibilities. This is a reasonable parallel to what is required of the computer-based systems developer who uses a predefined method. He or she can be seen as concentrating on using that method to develop his or her product, rather than modifying the method.

In summary, it is possible to reflect the major aspects of Kuhn's model of a discipline in the current state of CBSD.

4.2.2 Method Development and Pre-Science

The theoretical side of CBSD, that part devoted to the development of methods, tools and techniques for computer-based systems developers to use in designing, implementing and maintaining systems, can be considered as an analogue of a Kuhnian pre-science, because it shows many of the attributes of the pre-science. In particular, it is composed of many schools, each of which has:

- *Its own way of seeing the world*, which is embodied in the notations and modelling techniques advocated.

  Consider, for example, differing computational models: formal vs object oriented vs data vs process.[19] Each of these computational models can be used as the basis of a series of notations for describing problems or solutions. Different computational models underlying the notations available may cause the analyst or designer to see the world in different ways; the dataflow modeller will see flows of (passive) data to and from stores and being modified in processes; the object-oriented modeller will see numbers of objects, each with its own set of allowable actions, moving around.

  Does this difference result in any fundamental differences in the models produced? This is a question which needs to be asked and answered, in ways more formalised than the metaphysical appeals to 'naturalness' which are common in current literature.[20]

- *Its own set of beliefs and assumptions* concerning such fundamentals of CBSD as the process models to be employed (for example, waterfall vs incremental vs prototyping), and the importance attached to involving the user in systems development.

- *Its own definitions of key terms* such as 'object-oriented', on which members of differing schools of 'object-orientedness' are unable to agree, or on what constitutes 'quality' in a CBSD process or product.

In the context of the current state of theoretical CBSD, we must note Kuhn's criticism of the operation of pre-science disciplines, i.e. that progress is sapped by disagreements over philosophical bases (Kuhn, 1970, p 13) – is CBSD method and technique development suffering from the same problem for the same reason?

In seeking reasons for these strength-sapping divisions, we can refer to such real-world issues as different requirements of systems, different environments of operation, different constraints, and so on. We might also consider the non-system-related effects of the commercial world and of academic pressures, which

---

19 Sometimes these are mixed or combined in individual methods or notations.

20 "OOA - Object-Oriented Analysis - is based upon concepts that we first learned in kindergarten: objects and attributes, wholes and parts, classes and members." (Coad and Yourdon, 1991, p 1)

combine to press for product differentiation. How often does one hear a consultant saying proudly that his or her organisation has 'its own methodology',[21] regarding this as a selling point rather than as exhibiting a weakness in the discipline? This reflects an attitude in which difference *per se* is felt to be distinctive and good, rather than representing a failure to combine under a single theoretical umbrella as might be felt in a mature science. The effect of market forces must also be taken into account, requiring saleable, working answers before the relevant questions have been answered by theorists in terms agreed by all. This agreement can be seen to be expressed in other fields by standards set by bodies established for the purpose – but who would set the International Standard for, say, the object-oriented computational model, and how might a general agreement on that standard be obtained?

Those actively involved in developing a method or family of methods are not alone in espousing it. There is also the group of those active developers who use and believe in the method. In fact, these two groups might overlap since methods are often developed and innovations brought in by active computer-based systems developers, such as Michael Jackson, rather than by pure theoreticians. Action research programs neatly straddle this divide between theory and practice.

We can therefore identify a group of people, possibly much larger than the developers of a method, who use that method and are influenced by its belief system, possibly without understanding the theory underlying it. These people might be paralleled with workers in the chemical industry, manufacturing products according to mechanisms developed by the theorists, but without necessarily fully understanding what they are doing or why they are asked to do it. However it is possible that more creativity is needed in developing a computer-based system, albeit according to a predefined method designed by another, than following a recipe in chemical manufacture.

Despite the characterisation of CBSD as a parallel to a pre-science discipline divided into schools, I believe that it may be possible in the future to identify a common core of beliefs which characterises 'software engineers' as a group distinct from other computer-based systems developers, in the same way as adherence to a

---

21 An example of this was found in the interview (see Chapter 5, below) of subject K (transcript, p 7), in which he referred to his experience of "what would now be marketed as a methodology".

paradigm unites the workers in a scientific discipline, even if other beliefs divide them.

4.2.3 Method-based Computer-based Systems Development and Normal Science

> "To a man who has only one tool, a hammer, everything looks like a nail." (Confucius)

We can consider the practical side of method-based CBSD – actually developing and implementing computer systems – as being an equivalent of Kuhnian 'normal science'. There is often little questioning of the philosophical basis of the mechanisms employed during the process of developing a system under the control of a predefined method or set of techniques.

I suggest that the developers who take on a particular method, i.e. who believe in it as a valid method for developing systems, can be said to have joined a community associated with this method, and taken on the associated paradigmatic beliefs and models enunciated by those who defined that method. However this paradigm is filtered, *inter alia*, through the existing beliefs and models in the minds of the developers concerned. The DM of a CBSD method can be viewed as a cognitive filter, modifying a computer-based systems developer's view of the world (cf. Episkopou and Wood-Harper, 1986; Petre, 1989[22]).

It is interesting to note Kuhn's justification for this lack of questioning of bases in science – it allows rapid development of the practice of a discipline without the need to rebuild the philosophical basis each time (Kuhn, 1970, p 42). Rapid development without much debate on the philosophical basis of the method employed, once the method has been chosen, is what is needed to develop computer-based systems. It does not need further justification; it is the right thing to be doing. Just as the schools of a pre-science discipline spend all of their time arguing over or reinventing basic principles, rather than applying themselves to detailed work on a basis agreed by all, it might be easy for a computer-based

---

22 In support of the use of Petre's work on programming-in-the-small in this thesis which is concerned with larger-scale work, consider Jackson (1989) in the context of a critique of the concentration on 'languages' for expressing CBSD specifications and designs: "I want to argue precisely that it is in today's 'programming in the large' that the influence of the programming language paradigm can be seen to have its effect."

systems developer to spend all his or her time questioning the tools used rather than actually developing systems, a situation which would defeat the purpose of the tools themselves, as well as delaying the progress of the CBSD process.

The DM within which computer-based systems developers work, consists of elements derived from a number of sources: personal, organisational, role, project and discipline. It may be that beliefs or models derived from these source might conflict. At worst, disbelief must be suspended in order to apply the tools effectively. Additionally, two developers developing systems using tools built under differing methods, espousing different schools of thinking, will, according to this Kuhnian parallel, find the concepts they are using incommensurable in respect of the DM elements in which they disagree. This explains the need for a species of *conversion* when a developer changes his or her DM by changing to a new method based, for instance, on modelling the system using a different computational model.

The tools used by a computer-based systems developer influence how the circumstances of a situation are interpreted, what is seen as the problem in that situation, and what is a valid solution to the problem. There is a striking parallel between the effect of a Kuhnian paradigm on a scientist and that of these influences on a computer-based systems designer; both act as theory-based filters on the way in which the practitioner sees the world.

In considering the detailed parallels between the activities of method-based computer systems development and Kuhnian normal science, the following points therefore emerge:

- most importantly, as stated above, the filtering effect of the underlying belief system on the framing of problems and the definition of acceptable solutions to them;
- the parallel between CBSD within a method- or tool-based framework which constrains and controls it, and normal science, both operating within a paradigm which constrains visualisations of problems and definitions of acceptable solutions;
- the parallel between:

- the acceptance of a change of the set of assumptions behind different tools for an individual computer-based systems developer; and
- the gestalt switch for a scientist undergoing the religious conversion of a change in paradigm (Kuhn, 1970, pp 150–151); but consider the difference between persuasion and conversion below;

- the difference between *persuasion* of a person to a new theory (by convincing him or her that the new one is better in some way) and *conversion*, in which a person comes to believe in the new theory, see the world through its eyes (Kuhn, 1970, pp 202–203) and 'go native' (*ibid.*, p 204), parallels the different ways in which a person may use a new method, technique or notation. He or she might *know* that it is better in some way than his or her existing mechanism, but not really *see* the world through the new method's cognitive filter. We can compare this with the possibility of having a new DM element, disagreeing with his or her existing beliefs, forced on a developer; and
- the need for translation between different scientific communities, forced by the different underlying DMs (Kuhn, 1970, p 202), parallelling the difficulties in translating the ideas of computer-based systems developers into terms understandable by non-software professionals, who are not under the influence of the DM, during a development.

To summarise, the following effects are predicted by an application of the analogy of Kuhnian normal science to method-based CBSD activities. In general, developers will act and decide under the influence of the DM of the method adopted. In addition, developers may attempt, or be made, to act in accordance with the DM even if they do not believe in it, but this might result in some degree of *cognitive dissonance* (Weinberg, 1971, p 54-56).

4.2.4 Informal Development and Individual Developers

Here, the analogy with any aspects of Kuhn's work is less clear. A developer using *ad hoc* mechanisms will have no input from rigidly defined development mechanisms, since by definition these mechanisms do not exist.

This aspect has been implicitly explored in the investigations described in the next chapter of this thesis; individual and *ad hoc* developers have been included in the

interview subjects. The results to be presented later in this thesis will show that informal developers share some beliefs and models with their more formally-based brethren (see Appendix 2). These elements of a shared DM will influence the work of the informal developer in the same way as they do that of the method-based developer.

4.2.5 The Contexts of Philosophical Debate

Why should it be that method developers diverge in their opinions of the underlying beliefs in a method, whereas those who actually apply that method in developing systems are not expected to do so? I suggest that this is due to the different roles within CBSD of method developers and of method users.

It is the task of those who develop CBSD methods, either developing or extending methods and whether in industry or in the academic world, to question the underlying assumptions and models in a method, to examine it to see whether they can improve it in some way. There is no problem in their questioning the assumptions implicit or explicit in a method, because that questioning is an important part of their job.

By contrast those who develop computer-based systems are usually expected to adopt the tools provided and use them. The developers or their predecessors might have chosen the tools themselves, or senior management might have imposed them for any of a variety of reasons including a desire to control those whose crucial work they do not understand. The only alternatives to adopting and using the mechanisms adopted open to systems developers are either to change jobs, or to use the mechanisms half-heartedly, possibly working from within to undermine the mechanisms' effectiveness either deliberately or unwittingly.

4.3 The Analogy Described 1: Kuhn's Communities

Having outlined a parallel between Kuhn's theories and CBSD activities, I now consider in general terms how well the concepts embodied in Kuhn's work – scientific communities, the DM, and revolutions – fit into the current state of CBSD. Later in this thesis, I will be able to examine these issues in more detailed terms, informed by the investigative work which I have performed; for the present, I will look at these matters in outline.

4.3.1 The Sociological Nature of Kuhn's work

Kuhn's approach to his subject can be seen as being essentially sociological in nature. He explores the nature of scientific communities, and how they are affected by the emergence of, and changes to, the underlying DMs which characterise them and their members. His description of scientific revolutions, for instance, owes at least as much to their effects on the affected communities as to the scientific rationale for such changes in underlying belief system.

My approach to the work presented in this thesis has been similar to that of Kuhn. By applying Kuhn's model of a discipline to CBSD, I have assumed that:

- a community of people working in CBSD can be identified; and
- some aspects at least of a common DM can be identified, which unites the members of that community, or at least groups them into schools within the discipline.

The first of these assumptions seems reasonable in that 'computer-based systems developers' are identifiable by a well-defined activity which they all perform, i.e. developing computer-based systems. The second assumption will be explored in the next chapter of this thesis by investigations aimed at identifying elements of the DM of CBSD which are explicit and implicit in textbooks and practitioners' mind-sets.

4.3.2 The Scientific Community and its Parallel in Computer-based Systems Development

In describing his communities of scientists, Kuhn (1970, p 178; p 181) concentrates on the low-level communities of 25 to 100 people where scientific research really takes place. In considering communities of computer-based systems developers, we are talking at times about much larger groups. However, the community size for the groups who actually develop *and publish* details of new techniques, methods and so on may be similar to Kuhn's idea.

I identify the nearest analogue of a *scientific community* as the group of CBSD method developers and researchers who espouse and are developing either a common method based on a single DM or a family of methods whose relationship is defined by the closeness of their DMs. Such a group is close to a scientific

community in its shared belief in the answer to at least one fundamental question, *viz.* 'does our method work?' – the answer should be 'yes'.

Although the community-based aspects of the Kuhn-based analogy have not been explored in detail in the work described in this thesis, the definition given here of the equivalent in CBSD to scientific communities may result in a close match with Kuhn's expected size of a scientific community, of about 25–100 active researchers. However, the identification of such a community in method development may be problematical, since any 'community', identified by having a DM different from that of other communities, might be subdivided according to lower-level differences in elements of the DM, which, at a higher level, they might be seen as sharing. For example, the number of academic and industrial researchers into the nature and use of formal methods in CBSD is likely to exceed 100, but it is also likely that it will be possible to identify sub-communities of 25–100 by reference to the divisions within the formal methods community itself. The number of active developers of an individually-identifiable method, such as JSD or SSADM, might be closer to the bounds suggested by Kuhn.

If we are enumerating those who have adopted, or at least been exposed to, a particular method, we also need to consider the place of those developers who *use* that method. These people might be seen as followers rather than as members of the relevant community, taking on board, possibly partly consciously and partly unconsciously, the belief system underlying their chosen or adopted method. It is important to remember here that the reasons for a practitioner adopting a particular method are varied. For instance, the reasons may be personal research, the perceived stature of the method's developers, and/or diktat from above in a management hierarchy. For inclusion in what we might call the 'supporting community' of a method, the criterion might be whether the individual has adopted and internalised the DM of the method; as with Kuhn's mechanisms for the adoption of a DM, appeals to and pressures from outside the discipline, such as from management in a hierarchical organisation, may not be regarded as valid reasons.

The relationship of a computer-based systems developer to a method might be considered by an examination of what happens when a method changes, perhaps a new version being released and adopted. Do the developers who use it change their belief systems to match changes in the new method's DM for the same reasons they adopted the old version, or is there sometimes resistance to any new

beliefs introduced in the new version? Investigations into the effects of such is suggested in the 'future work' section in the concluding chapter of this thesis.

In order to differentiate between the Kuhnian scientific community and its equivalent for CBSD, it may preferable to refer to the latter by a term taken from semiotics, the *thought community*. I use this term, shortened to 'community', in the remainder of this thesis.

4.4 The Analogy Described 2: The Disciplinary Matrix

4.4.1 'Normal Science' and Computer-based Systems Development Practice

I have previously drawn a parallel between Kuhnian normal science with the use of a CBSD method, or member of a family of methods whose relationship is defined by the closeness of their DMs, by computer-based systems developers. To Kuhn, the function of normal science is developing the details of a theory within the underlying DM. Can this be considered to be equivalent to 'normal' computer systems development, i.e. developing systems within the constraints of a predefined method?

As I have suggested previously, it can be seen to be advantageous for a computer-based systems developer not to question too deeply the assumptions behind a CBSD method, technique or notation in order to get on with the detailed work of developing computer-based systems. I suggest that not only is it advantageous, but, in a manner analogous to Petre's (1989) observations on computational models of programming languages, the DM underlying the CBSD method employed forms a part of the view which a computer-based systems analyst or designer has of the world and the system.

What is the effect on this parallel of the common practice of tailoring CBSD methods for specific organisations or projects? Can we parallel here both the use of a method unaltered, and the use of that method modified to meet either the circumstances of one project or of all projects for or by a particular organisation, with scientific research carried out under a single DM? It should be noted that some methods, for example SSADM version 4 (Downs *et al*, 1992), specifically include in their mechanisms the ability to be modifiable, so that they already carry with them the belief that modifiability is an advantage in a method. Other methods are more rigid and prescriptive, setting out the tools to be used and the

process in which they are to be used; I would include Structured Design (Constantine and Yourdon, 1979), as originally expounded, in this category.

Perhaps the best answer is that some DMs underlying CBSD methods allow themselves to be stretched in certain directions and this is reflected in an expectation of flexibility in the methods devised using these DMs as a starting point, whereas other DMs are more rigid. The degree of modification possible before a method becomes sufficiently different from the initial version for significantly different results to be expected due to changes in the influence of underlying beliefs and models is a matter for future research.

Kuhn (1970, p 35 *et seq.*) regards normal science as puzzle-solving within the limits set by the DM. This reflects the way in which the DM sets the problems of normal science as well as providing information on how to recognise 'good' solutions to them. In a design discipline, the requirements are often set from outside the discipline, for example by clients or users of computer systems to be developed. This is another example of the difference between a science and a design discipline. However, developing computer systems can still be thought of as puzzle-solving within the limits set by the tools, models and beliefs used, which will define how to build solutions and what constitutes a 'good solution'.[23] The tools are themselves reflections of their designers' underlying DMs, and to that extent CBSD work using predefined tools can be seen as puzzle-solving within a paradigm. The system developed can be seen as an embodiment of the hypothesis that 'this solution meets the problem as previously defined.' This hypothesis is sometimes not explicitly stated, except perhaps in an invoice for work completed, and is rarely justified in theoretically sound terms. An exception to this might perhaps be a system defined using formal methods to meet a specification which has been set out exclusively in the notation of that formal method.

Can 'normal science'-like thinking be observed in design disciplines other than CBSD? Gordon (1978) provides an example, from the academic area of structural engineering, of the mind closed to new ideas which seems to be a part of the rigidity engendered by 'normal science' (*ibid.*, p 260):

> "Even as late as 1936, the basic Lanchester-Prandtl (or vortex) theory of fluid dynamics was neither taught nor permitted to be used in the Department of Naval

---

23 The place of the values held by the developers is particularly significant here.

> Architecture in the University of Glasgow ... much the same thing happens with 'modern' theories of fracture dynamics in present-day engineering departments."

This example cuts across the argument that modern engineers are fundamentally practical thinkers, gladly accepting any tools which would help them to get the product right. The quotation shows 'normal science' at work – if it does not fit into the current DM, it can (and will) be ignored. Kuhn's work forms the philosophical basis for an explanation of the conservatism of engineers.

It can also be noted that a scientific theory is also a designed artifact, attempting to explain observations with the confines of a DM. To that extent, the scientist is also a designer.

4.4.2 The Unification of a Discipline under a Single Disciplinary Matrix

In the work presented here I do not make any claim that the ultimate result of work in developing CBSD methods is, or should be directed towards, the emergence of one method or DM which replaces all others in the emergence of an analogue of a Kuhnian 'science'. Claims of this sort are made by some developers of methods, but I am not judging the validity of such statements. By adopting Kuhn's model to describe CBSD, I am not stating that the eventual outcome of methodological developments must be a discipline unified by a single DM as is a Kuhnian 'science', neither am I suggesting that such an outcome is impossible.

Kuhn (1970, p 17) notes that it may be that the elimination of competing schools and the emergence of a single unifying DM may be unique to a science. I therefore suggest that for a field which is not necessarily a science, such as CBSD, the continued existence of competing schools into the maturity of the discipline is not contradicted by Kuhn's work.

4.4.3 The Subdivision of the Disciplinary Matrix

As noted previously, Kuhn divides the elements forming the DM of a science into symbolic generalisations, beliefs, values and exemplars.

As will be seen in Chapter 6 below, this division in the DM of CBSD can be supported by evidence from my investigations, but the analogy is only partially successful. In particular, there is a dearth of symbolic generalisations. This may

be due to any or all of a number of possible reasons, including the difference between a scientific discipline and a design discipline, or particular peculiarities of CBSD, or the current comparatively early stage of development of CBSD as a discipline.

4.4.4 Lack of Criticism of the Paradigm

Kuhn (1970, pp 45–46) notes that, in a normal science period, scientists are uncritical of, and to some extent unaware of, the DM within which they are working. Chalmers (1992, p 93) later comments that a scientist is unaware of and unable to articulate the paradigm because of his training within the paradigmatic culture.

The source of such an uncritical acceptance for a scientist can be paralleled with either or both of the training of computer-based systems developers, often by learning and using a particular method (see, for example, Sommerville, 1992), and, by drawing on a parallel from investigations at a smaller scale, the work on 'head languages' by Petre (1989). However, whether such an uncritical acceptance is true of computer-based systems developers, or whether such aspects as the desire to modify methods to meet particular circumstances as noted above, reflects a less rigid acceptance of received wisdom, remains to be investigated. The question to answer in future work is that of whether or not modification of a method implies criticism of the underlying DM.

However, as noted previously, this uncritical acceptance is only applicable to the practical side of the discipline of CBSD, i.e. the development of computer-based systems, rather than to the development of the methods which practitioners use.

4.4.5 The Place of Textbooks

For Kuhn, textbooks form an important aspect of a science. They encapsulate explicitly or implicitly the DM of the time, relate an historic context in 'modern' terms, and present exemplars for study and emulation.

The textbooks (and occasionally papers, although the details of the methods are more commonly described in books and the latter are the common source for practitioners) in which modern CBSD methods are described serve the same

purposes. These books certainly encapsulate the DMs of their authors, and exemplars are provided for study, and as test exercises and/or solutions.

The possibility of examining textbooks for evidence of the rewriting of history to conform to more modern paradigms is considered below, and included in the future work section to this thesis.

4.5 The Analogy Described 3: Are There 'Revolutions' in Computer-based Systems Development?

4.5.1 Pressures for Crises and Revolutions

I consider here in outline the pressures which might result in an equivalent to a scientific revolution in CBSD.

To Kuhn the function of a scientific revolution is to allow the bounds of a DM to be broken when required to advance the theory of a discipline in the face of unacceptable pressure on the existing DM. What might constitute these pressures, which force a revolution in a science, for the discipline of CBSD? What would cause such a sense of crisis and scandal? Certainly the current state of CBSD, with the number of publicised failures to meet specifications, budgets and/or deadlines, might imply that if it is not in a state of crisis and revolutionary activity, then it ought to be.

Examples of the potential pressures on the discipline of CBSD include the need to advance its practice in the face of new demands such as:

- greater cost and/or time constraints;
- the need for better functionality;
- the requirement (whether real or imagined) to adopt new technology as it arrives, such as 4GLs or the object-oriented computational model; or
- the need to work within new technical and/or commercial environments.

It should be noted that these pressures, as well as the sense of crisis, come to some extent from outside the discipline, rather than from inside as Kuhn expects for a science. This may reflect another difference between CBSD and Kuhn's vision of a science, in that Kuhn believes that only the practitioners in a discipline can

determine the results of revolution by comparing the competing paradigms, whereas in CBSD the stake of wider society is such that others can and do control the pace of change by means of the pressures on the practice of the discipline.

At a personal or organisational level, a crisis might be induced from outside the local community of computer-based systems developers by the introduction of a new method. For example, we can consider the role of management in changing the method used, convinced of the superiority of a new method either by its adherents outside their organisation or by their own specialists. Does the crisis occur in part in their heads, rather than those of the computer-based systems developers? Alternatively, the dissatisfaction of one or more members of an existing CBSD school's community, perhaps in the same organisation, with their existing method, and implicitly or explicitly some aspects of its underlying DM, may provoke a change in the tools used. This local effect can be distinguished from a general unease throughout the community of all computer-based system developers, most of whom will continue to use their existing methods, with their DMs, quite happily.

It should be observed that, to Kuhn, crises and revolutions are characteristics of a science. I have already described the similarities between CBSD practice and 'normal science'. Revolutions might therefore be expected to occur within such an environment. I have also suggested previously that I believe that the theory-building activity of CBSD, resulting in the development and publication of new or modified methods, is in a pre-science stage. However, looking into the earlier history of CBSD, one discipline-wide example of the sense of crisis followed by and engendering change as suggested by Kuhn might be that of the events leading to and following the Garmisch conference (Naur and Randell, 1969), where many concerns were voiced about the then-current state of CBSD, based on scaling up programming mechanisms, and the term, if not the discipline, of Software Engineering was invented.

It is also reasonable to ask how many of the changes and advances in the theory and practice of CBSD have been the result of crisis and revolution rather than the step-by-step, gradual adoption of new mechanisms alongside the existing (and continuing) ones.

4.5.2 Paradigm Shifts: Additional Influences for Change

I set out here some of the influences suggested by Kuhn on change from one DM to another, together with examples which Kuhn has used to illustrate each. I then comment on possible parallels in CBSD.

My comments on each guideline are necessarily brief at this early stage of development of my work using Kuhn's theories. I suggest that further research could be profitably undertaken into each of these aspects and its effect on change in CBSD theory and practice, and have therefore included this in the 'future work' section of the final chapter of this thesis.

Religion (Kepler's sun worship - Kuhn, 1970, p 153)

If in 'religion' we include elements of practitioners' belief systems which are more fundamental to them than any part of the discipline knowledge, and which are perhaps based more on faith than on (pseudo-?)scientific rationality, there may be a correlation.

Prior reputation of the innovator and his teachers (Kuhn, 1970, p 153)

This undoubtedly has an input into the process of selecting a CBSD method or technique.

For instance, B. Martin (1994, p 21) deems it to be of sufficient importance to mention the fact that James Martin developed the RAD technique, although this is disputed by some.[24] For B. Martin, and those expected to read the document, the name 'James Martin' is a recognisable talisman for acceptability and goodness of the product in question. The reputation of the teacher is transferred to his teachings.

An example from practice of the same craft master's reputation being invoked as a touchstone of validity of a method comes from interview subject J.[25]

Nationality of the teacher (Kuhn, 1970, p 153)

The nationality of the authors of a method, or at least the country in which it was developed, seems to have an effect on the method itself. This could be thought of

---

24 R. Winder, personal communication, 1995.

25 For a description of the interview-based investigation, see Chapter 5 below.

simplistically as the 'not-invented-here' mentality writ large. However, more complex cultural and legal issues are involved.

Consider the 'Scandinavian School' of user-centred design (see, for example, Groenbaek *et al.*, 1993), whose practices are greatly influenced by the need to work within a society which has decided to make it a legal requirement to obtain consent from representatives of the end users to the implementation of a computer system. Under these circumstances, it is more likely that a system under development will have to be acceptable to the users, and CBSD methods must take this requirement into account. Here we see outside influences at work, namely those promoted by society as a whole, influencing the results of computer-based systems developments.

This is one area in which the analogy with Kuhn's work is weak, in that Kuhn believes that those outside a science have no influence on the acceptability of a particular paradigm to those working in that science.

Nature of the teaching system (Kuhn, 1970, p 165)

Kuhn's view of the teaching of science is that it relies to a great extent on textbooks written specifically for student use, as against the more critical, open mechanisms used in the arts. Original scientific works, such as those of Newton and Faraday, are replaced in the science curriculum by summaries prepared in a modern context for pedagogic purposes.

This style of instruction is certainly also applicable to CBSD teaching. The number of textbooks currently being produced is vast, but students are not encouraged to read and critique documents of the history of the discipline, such as the proceedings of the Garmisch conference (Naur and Randell, 1969).

It is interesting to note that the current mechanism of teaching CBSD may enhance the invisibility of any revolutions which may have occurred in the past, if the rewriting of history in newer textbooks suggested by Kuhn has occurred.

Reference for decisions can be made to domain experts only, not outsiders (such as political leaders, the general public – Kuhn, 1970, p 168)

> "One of the strongest, if still unwritten, rules of scientific life is the prohibition of appeals to heads of state or to the populace at large in matters scientific." (Kuhn, 1970, p 168)

In the development of computer-based systems, there is a definite difference here between Kuhn's 'science' and CBSD. There are undoubtedly pressures on computer-based systems developers from outside the CBSD community. There may be a more prescriptive legal framework surrounding a computer-based systems developer, such as the Data Protection Act, the Scandinavian workers' rights legislation referred to above, and U.S. product liability legislation, to which the developer must conform, or which have an influence on what the system is to do, and/or on how a computer-based system is to be developed. Contractual issues also affect how software products are developed, as well as the products themselves.

However, this issue may be treated differently on the theoretical side of CBSD. Discussion and resolution of technical issues related to the development of methods for developing computer-based systems are carried out at conferences and in the literature, and methods are modified by CBSD practitioners. These discussions are not allowed out of the domain of 'experts' within the discipline. New and borrowed ideas are imported by workers in the discipline – I instance the work presented in this thesis – but acceptability is finally decided by method developers and users.

The failure to appeal to the public might even be seen as a source of problems in CBSD, as in cases when the results of design projects given to their users are sometimes, and very publicly, rejected by users or otherwise found to be inappropriate.

The philosophical theories of science which I have considered for this thesis also ignore the need nowadays for scientific workers to persuade non-scientists to espouse, or at least finance, the work required to advance a discipline. Is this not now becoming as vital as the need for a scientist to convince his peers? Is not now the successful theory that whose adherents can obtain funding for experiments rather than that which commands greatest respect or hope for advance from advocates' peers? Dasgupta (1991, p 11) considers this matter in the context of the funding agencies influencing research priorities for their own reasons: "...the sanctioning, by a funding agency, of a research topic that it may wish to support may serve to *define* which problems are important and which are not. One may, thus, see the insidious influence of the agency's particular biases on what the relevant research community *believes* to be important."

It appears therefore that, even in science, it is not science alone which determines the success of a scientific research project, but the wider political and/or financial influences which underlie the practicalities of being able to perform the necessary work. CBSD research is further constrained by the needs of the community of CBSD practitioners, who demand methods and tools which allow them, in turn, to meet the needs of those who pay their wages. The difficulty of providing methods and tools of acceptable performance does not reduce this demand for such tools.

The new theory shows an ability to solve problems not attacked by the earlier one, and not re-open too many already-solved problems (Kuhn, 1970, p 169)

Over time, the problems considered by the majority of computer-based systems developers have changed. Systems designers are insulated from the lower levels of hardware interaction, can use sophisticated data handling models and tools, and have a rich diversity of user interaction possibilities. The problems which they are asked to resolve have become more complex, and the tools and equipment with which they are to solve them more capable. Therefore it is undoubtedly the case that new methods, and in particular computer-based tools, can address problems with which earlier methods would have struggled, if only because the earlier methods did not take account of more recent, more complex, development environments.

The question of re-solving existing problems can be considered under two headings:

- *re-engineering or replacing existing computer systems:* this is performed using newer, 'better' tools, to replace systems which are no longer usable due to non-portability, unmaintainability, or other causes; and

- *reinventing the wheel:* this is the case of same problem in a different organisation.

I am unaware of any existing, solved problems in CBSD which have been rendered insoluble due to the application of new methods. Although it is unlikely that the proponents of these methods would have publicised failures of this kind, I suggest that such problems are in fact unlikely to occur.

Novelty by itself is not a criterion in the sciences for adopting a new theory. (Kuhn, 1970, p 169)

Here we see some another area of divergence between CBSD and Kuhnian science. A new (or modified, repackaged, or renamed) CBSD method may be adopted for reasons other than it being a better source of solutions.

For instance, commercial pressures may demand a unique method for a consultancy, or a 'new' product to sell for a systems development software tools publisher. An updated method requires new textbooks, software tools and courses. Academic pressures require a stream of publications, and 'new' methods and techniques are a source of these. The interest in, and commercial pressures to change, CBSD *mechanisms* is a point of difference between the history to date of the development of CBSD and that of other design disciplines, in which such pressures have not arisen.

"There is no neutral algorithm for theory choice, no systematic decision procedure which, properly applied, must lead each individual in the group to the same decision" (Chalmers, 1992, p 108; quoting Kuhn, 1970, p 200)

Note particularly Kuhn's (1970, p 163) statement that:

> "Scientific progress is not different in kind from progress in other fields, but the absence at most times of competing schools that question each other's aims and standards makes the progress of a normal-scientific community far easier to see."

CBSD has precisely the division into competing schools that Kuhn refers to here, if we consider the as such the adherents to particular families of methods. I suggest that the 'choice' of DM is implicit in the choice of a method which embodies it, and the influences on this choice are, as suggested above, numerous and in part subjective.

4.5.3 The Incommensurability of Paradigms

As noted previously, Kuhn claims that different paradigms cannot be compared directly, because they use different terms or, more dangerously, they use the same terms with different meanings. He refers to the need for translation between different scientific communities, which is needed to resolve difficulties,

misunderstandings and ambiguities resulting from the different underlying DMs (Kuhn, 1970, p 202). Can we see any evidence for this occurring in CBSD?

One example for which it is difficult to measure the DMs underlying different methods against each other is that of the computational models of notations. Adherents of procedural, data, entity or object-oriented modelling notations would find it impossible to compare their models without some measure of translation and reinterpretation. As we have seen, there are claims of 'naturalness' for object-oriented modelling techniques, but to what extent to they reflect the comparative ease of use of that computational model when compared with any other which its user has already internalised?

I suggest that two different modelling notations may in fact be incommensurable, i.e. that it may not be possible for models written in the differing notations to be directly compared in all of their particulars, if the notations either describe different sorts of *things* and/or describe different sorts of *actions*, i.e. if their underlying computational models differ. If these underlying languages which the notations reflect differ in any respect, then, when models described in the two notations are compared, a thing or action which is not common to both will have to be translated in order for it to be understood in the language which lacks it. The two notations are not directly comparable in all respects without the constructions of circumlocutions in one notation to reflect concepts unique to the other.

What, however, about CBSD methods which use more than one notation in the same development? An example of a CBSD method which contains a number of notations, which, I claim, are not directly commensurable in all of their aspects, is OMT (Rumbaugh *et al.*, 1991). This method includes three modelling notations, which describe objects (the "object model"), states (the "dynamic model") and data flows (the "functional model"). I suggest that, were these notations to be identical in all of their aspects, then it would only be necessary to use one of the three to capture all of the information which all of them could provide.

How are the notations linked in a multi-model method to provide a unified picture of a design? In OMT, overlaps between the elements of the underlying computational models are exploited to associate each type of model with the others. Object and dynamic models are linked by the common concept of an *object*. The object model reflects static relationships between objects, whereas the dynamic model describes states which one object or a number of associated objects

can assume and the events which cause transitions between these states (*ibid.*, p 110 111). Functional models are linked to object models since the objects perform the actions which are the leaf nodes of the functional model (*ibid.*, p 137). Data stores in the functional models are objects, and data flows are values, in the object model, (*ibid.*, p 138). The functional model describes actions whose sequence is described in the dynamic model (*ibid.*, p 139).

Despite their demonstrated ability to combine models written in differing notations into a single design, the insistence of Rumbaugh and his colleagues on the employment of three notations implies that these researchers see each notation as containing some aspects of the complete system which the other two cannot capture.

All of this is not to say that a number of notations cannot have many common elements in their underlying computational models. At the extreme, two notations may be directly commensurable if they reflect exactly the same computational model and the only differences between them are, say, the shapes of graphical notational elements. A direct translation of a model from one notations to the other would then be possible, merely by redrawing the boxes.

Looking for different uses for the same term, we can see clearly different interpretations of the term 'object-oriented', reflecting differing views of what the term means and the effect of adopting such an approach.

Are the advocates of the differing method families, and therefore of their underlying DMs, really talking at cross purposes? Not necessarily – because of the many-faceted structure of the DM of CBSD proposed later in this thesis, many of the concepts will be shared between all practitioners. In the areas in which they differ, the results of their activities may indeed be incommensurable. The nature and significance of these differences will need to be explored as a real-world investigation following on from the work presented here.

4.5.4 Revolutions and Paradigm Shifts in Computer-based Systems Development Theory

I can observe no parallel at the discipline level of method developers to the replacement of one DM with another, after a scientific revolution, since the Garmisch conference referred to above. There is no sign of one method family,

based on a single DM, replacing all others, or of some being rejected as completely as is a superseded paradigm. This reflects the pre-science state of the theoretical side of the discipline.

However, the term 'paradigm shift' has become a commonplace among CBSD theorists, or at least those who market tools to developers, and is used as a label for any change in thinking. Many of these alleged shifts may indeed be paradigm shifts in Kuhnian terms, but until we have found and documented at least some of the elements forming the DM it will be difficult to tell how significant these changes are. Additionally, observation of the effects of such changes on CBSD theory and practice demonstrates that such changes are not universally accepted, and as a result the discipline, or a school based on a greatly-changed method, suffers from what might be more accurately called *paradigm splits* than paradigm shifts.

It should also be noted that in Kuhn's view it takes a generation for a change in DM to become complete, when the last adherents of the old paradigm die out (Kuhn, 1970, p 151). CBSD is still too young a discipline for this process to have been observed to completion, and we will have to wait for the perspective gained by a longer timeframe to check for occurrences of this phenomenon.

4.5.5 Revolutions and Paradigm Shifts in Computer-based Systems Development Practice

If an activity is considered to be equivalent to Kuhnian normal science, as I have suggested is the case for CBSD practice, some revolutionary activities might be expected to be observed.

As a hypothetical case, let us consider a systems analyst experienced in Structured Analysis and Design (Yourdon and Constantine, 1978). He or she models the real world and the system which he or she is creating in terms of data flows and of structures of program modules. Let us now train this analyst to use an object-oriented method, which requires him or her to model the real world and target system in terms of objects, each containing data elements and the process elements which act on them. The analyst's behaviour has changed to the extent that he or she now goes out into the real world to locate objects rather than identify individual, separate data flows or process elements. Let us further assume that the training or subsequent experience with the method does result in some measure of

replacement of his or her mental model of the world with a new one.[26] Internalisation of this new view of the world would result in him or her accepting a different DM, containing a mixture of existing elements, i.e. those unchanged by the change in computational model, and new elements which reflect the new modelling mechanism. We can compare this to the changes in world view which Kuhn (1970, Chapter 10) has suggested to have happened to astronomers and electrical experimenters.

I suggest that the changes described might result in the analyst, as the old method is forgotten and the new one continues in use, looking at the world in a different way, as do scientists after a revolution:

> "...the data ... collect[ed] are themselves different. ... Confronting the same constellation of objects as before and knowing that he does so, he nevertheless finds them transformed through and through in many of their details" (Kuhn, 1970, pp 121–122).

In our example, the passive data element, previously shunted around by active processes, has now gained a life of its own by becoming an object. The analyst has undergone what we might call a 'personal paradigm shift'. This forms a parallel to Kuhn's account of an individual scientist's conversion to a new paradigm.

That many of the skills, practices and procedures of the previous method might be carried forward to the new method does not invalidate the argument for the equivalence of the shift of a computer-based systems developer's DM to reflect a new method and a minor, personal 'scientific revolution', remaining within the discipline of CBSD, since this also occurs after a 'real' scientific revolution.

> "...postrevolutionary science invariably includes many of the same manipulations, performed with the same instruments and described in the same terms, as its prerevolutionary predecessor. If these enduring manipulations have been changed at all, the change must lie either in their relation to the paradigm or in their concrete results. I ... suggest that both these sorts of changes occur." (Kuhn, 1970, p 130).

---

26 Petre's (1989) work on computer programmers suggests that the internalisation of a new computational model is in fact never complete.

Perhaps the crucial difference between a scientific revolution and the change in a systems analyst's mind-set is that Kuhn suggests that the scientist who has experienced a change in view consequent on a scientific revolution sees his earlier beliefs as wrong, and his new ideas as a correction or replacement of that previous view. The analyst, unburdened with the need to consider one view 'right', and others 'wrong' or at least less 'right', may be able to attempt to move between the viewpoints provided by differing methods as required. However, such a repeated move may be impossible to achieve in reality, particularly for cases in which there is a great difference in viewpoint or when articles of semi-religious faith are involved, such as those explicitly espoused by the advocates of declarative as against imperative programming techniques (cf. Petre, 1989).

4.5.6 The Invisibility of Revolutions

Kuhn (1970, p 136 *et seq.*) describes the process by which scientific revolutions become invisible after their occurrence, specifically the rewriting of textbooks into the language of a new paradigm making the change from the old one invisible, since the historical introductions given in them tend to view the past as a smooth course towards the current position (*ibid.*, pp 137–140), and old ideas are rewritten in the language of the new paradigm (*ibid.*, p 138).

It is only by reviewing textbooks of the past that a perspective on the changes made in the theory and practice of computer systems development since its inception can be obtained. It is in this spirit that, for instance, the proceedings of the Garmisch conference (Naur & Randell 1969) might be read and examined in the context of current thinking in CBSD, as revealed by textbooks to show what is taught, and practical method 'cookbooks' and interviews with practitioners reflecting CBSD practice to show what is recommended for use and actually used. I suggest in the future work section of the concluding chapter to this thesis that such examinations of the history of CBSD be performed.

There is undoubtedly an advantage in applying this mechanism to CBSD, since the theory is well documented in books and other publications, and the discipline is comparatively young, allowing many of its pioneers to be interviewed.

4.6 Some Questions Concerning the Analogy

I set out here some questions which arise in making the analogical link between Kuhn's model of a science and CBSD.

Will the effect of the DM underlying a CBSD activity be perceived by the users of the software product?

Will the idea that the DM of a CBSD method affects the products of that method be reflected in the externals of computer systems, and, if not, where will it emerge?

If we accept Kuhn's view of what a DM is and implies, and if we further suggest that such a DM exists and underlies CBSD theory and practice, we are postulating that a set of beliefs exists, of some of which an analyst or designer might be unaware, which colour his or her professional actions. The DM forms a cognitive filter (cf. Dasgupta (1991) on the effects of the design paradigm), which will affect all of his or her work on a system, including external aspects and internal design decisions.

That the computational model(s) of the notations employed in a method affects the internal structural design of the resulting systems is trivial to show. An object-oriented method, producing object-oriented designs, will certainly result in a form of object-oriented implementation, even if the language used is not directly suitable (see, for example, the advice given on implementing object-oriented designs in procedural languages in Rumbaugh *et al.* 1991).

Where, however, is any evidence that the external aspects of a design are affected in a analogous way? Might there not be a greater effect on this derived from, for example, information and attitudes gathered from the users, an influence which might be independent of the way in which this information is gathered? Is it not more significant to determine the effectiveness of the information capture than the details of the mechanism which does it? I feel that we may not yet even be able to hypothesise that the DM governs external aspects of a computer system in any particular way.

Does the progression from one methodological DM to another follow the criteria which Kuhn sets out for determining which is 'better'?

Some of the influences on a scientist in selecting between competing paradigms are not scientific. However, Kuhn (1970, p 152 *et seq.*) claims that the following form part of the 'values' of the community in question, in determining the relative merits of DMs:

- success in solving the problem which provoked the crisis;
- accuracy of prediction, particularly of quantitative prediction;
- the prediction of new phenomena which can be subjected to experimental research;
- the number of problems solved; and
- less importantly, and more subjectively, aesthetic 'elegance', simplicity, scope and compatibility with other specialities.

Do the exponents of new methods implicitly or explicitly claim these attributes for their inventions, and how do they support their claims?

Forming an answer to this question is part of the 'future work' described in the concluding chapter to this thesis. An examination of the criteria by which one method replaces another, informed by the elements of the DM set out in Appendix 2 to this thesis[27], would provide useful information as to the true benefits, and the hidden effects, of such changes.

Will it be possible to develop as complete a theory for CBSD as there is for, say, physics? Is a theory for CBSD possible which provides quantitative predictions for the details of the outcome of a particular CBSD activity before that activity commences?

It is first important to note two differences between physics as a body of quantitatively predictive theory, and a predictive CBSD theory which this question implies might be developed. These differences are that problems of CBSD are both more complex, and less strictly defined, than are the subjects of the individual predictive theories of physics such as the motion of a particular body or the probability distribution of an electron. However, even were we to limit ourselves to the consideration of the possibility of a quantitatively predictive theory of CBSD

---

27 I particularly instance the influence of fashion; see Chapter 6, below.

relating to a specific system design to meet a single predefined specification, i.e. meeting a given specification rather than defining and then meeting that specification, I suggest that the answer to the question of the existence of such a theory based on our current knowledge of the discipline is 'probably not at present'.

The theories for some of the candidates for underlying scientific disciplines, such as sociology, psychology of programming, and theory of design, are not yet sufficiently well developed to support such a predictive theory for CBSD. We may, however, still be able to begin to identify or illuminate the driving ideas of CBSD and gain an initial understanding of the discipline, even if that theory is currently unable to match the physical sciences in predictive power.

4.7 Three Caveats

I note here three matters to be taken into account when considering the applicability of Kuhn's theories to CBSD, and the way in which the work presented here has been performed. These are:

- *Incompleteness of effects:* not all of the effects which might be predicted by the model due to the effect of the DM on CBSD practice can be expected to be clearly visible and perfectly applied in any single instance of that practice, because, as will be seen from the number of elements found for the probably incomplete DM presented in Appendix 2, the influences of the DM on CBSD practice are many and complicated, and the results of interactions between these elements are as yet unexplored.
- *The application is pragmatic:* the work presented here is not based on a 'scientific' (for instance, a psychology-based) approach, and a complete theory to underlie it is still lacking. I have not used, or attempted to build, a psychology of CBSD by analogy with the existing work on the psychology of programming. I am relying on a pragmatic application of an analogy with Kuhn's view of a science.
- *Not a science:* I am making no statement in applying this analogy that CBSD can be considered to be anything like a 'science', employing the latter term either as generally used or in Kuhn's terms as a discipline united around a single DM.

Chalmers notes that "[Kuhnian] Normal science involves detailed attempts to articulate a paradigm with the aim of improving the match between it and nature." (Chalmers, 1992, p 91). In particular, it can be noted that there are major differences of objective between scientists and computer-based systems developers. Scientists are looking for *laws of nature*, rational generalised predictive assumptions about the world in general. The computer-based systems developer, in particular the developer of bespoke software, is looking for a solution[28] to a single real-world problem, quite possibly without the wish to generalise it, and certainly without intending to raise it to the status of a general 'law of nature'. CBSD is a design discipline in Long and Dowell's terms rather than a science. The parallel I draw here between developing computer-based systems and Kuhnian normal science must therefore be restricted to the common idea of an identifiable community of people working within the limits set on their imaginations by a shared belief system, the DM, rather than by any shared type of objective.

4.8 Possible Sources for Elements of the Disciplinary Matrix

It is useful to note at this point that the DM which underlies the work of the computer-based systems developer when making decisions during his or her work is a combination of beliefs and models from many sources. Just as Kuhn notes that influences such as the nationality of the teacher might affect the scientist's decision as to whether to adopt a particular DM, so there are inputs to the DM in force at any point from various sources.

These sources of influences on the developer include the following. Note that not all of these might apply in any particular development situation, especially for cases in which formalised development mechanisms are not employed:

- society as a whole;
- the discipline of CBSD;
- the organisational context within which the work is being performed;
- the domain of application of the system;

---

28 Not *the* solution, only one which is acceptable; compare Simon (1975) on the nature of satisficing processes and solutions.

- the role within which the person is acting – analyst, designer, manager, etc.;
- personal beliefs of the computer-based systems developer;
- the method employed;
- the techniques, tools and/or notations used, whether they come from the method employed or are added on afterwards; and
- the support tools, such as CASE tools, employed.

The area addressed by the work presented in this thesis has been deliberately limited to that of the discipline of CBSD, i.e. those elements of a DM arising from methods, tools, techniques and notations.

Interactions between beliefs and models from these sources will be considered in Chapter 7, after the Kuhn-based model of CBSD set out in this chapter has been extended in the light of the practical work described below.

4.9 Summary

In this chapter, I have described how Kuhn's models of pre-scientific and scientific disciplines can be applied to describe the current states of the theory and practice of CBSD respectively.

In succeeding chapters, I will describe the investigative work which I have performed to test the validity of the use of Kuhn's work, and the more sophisticated application of his work which these investigations have allowed.

5 Some Investigative Work

> 'If a thesis has to be maintained which purports to be "practical," and to chastise the tendency to abstraction, that thesis is best maintained by a continual appeal to fact.'
>
> (Hillaire Belloc; Introduction to two-volume Everyman Edition of 'The French Revolution'; Thomas Carlyle; J. M. Dent & Co., London, 1906; vol 1, p viii)

5.1 Introduction

This chapter describes the investigations which have been performed to explore the validity of the Kuhn-based model of CBSD. This work has consisted of a search for elements of the DM of CBSD through the following sources:

- textbooks;
- transcripts of interviews held for the purpose with a number of practicing or former computer-based systems developers;
- a book describing the results of an action research programme into a framework for CBSD; and
- a report into the failure of a high-profile CBSD project.

Following this, conclusions are drawn as to the general validity of the analogy between the Kuhnian model and CBSD, and the specific parallels between Kuhnian pre-science and CBSD theory and between Kuhnian normal science and CBSD practice.

5.2 Introduction to the Investigative Work

This section outlines the aspects common to all of the investigative work carried out and described below.

5.2.1 Rationale for the Investigations

5.2.1.1 Objectives of the Investigations

Stated in the broadest terms, the hypothesis examined by the investigations in this chapter is that the Kuhn-based model of CBSD is reasonable, and that:

- an identifiable DM of CBSD therefore exists; and
- some, at least, of the elements forming that DM can be identified using informal mechanisms.

The overall objective of this work has been to provide evidence concerning the existence of the DM, and to begin the work of populating it with elements which have some measure of evidence to support them. More formally-based experiments which would build on this hypothesis are outlined in the future work section of the concluding chapter to this thesis.

5.2.1.2 What to Look For

An examination of Kuhn's work suggests three aspects for which it might be straightforward to look for evidence, *viz.* the community of practitioners, the DM, and some parallel with scientific revolutions. It is trivial to observe that there is a community of people who develop computer-based systems, and a parallel community of those who develop methods, techniques and tools for the development of computer-based systems. They can be identified by what they do. However, this does not advance the analogy, nor does it help us to identify or understand the effects predicted by Kuhn, such as the effect of the paradigm on practice or the incommensurability of paradigms.

I have discussed previously the possibility of revolutions in the current state of CBSD. It may be possible to find evidence for some revolutions over the history of the discipline of CBSD, but according to Kuhn himself these revolutions may be invisible in retrospect, and may therefore be difficult to identify unless we place ourselves in the positions of those who actually took part. Additionally, Kuhn's view that a revolution can only be complete after the last adherents of the 'old school' have died out reduces our chances of seeing one in such a young discipline. An examination of the history of CBSD for revolutions and changes in DMs is proposed in the conclusion to this thesis as future work, but is suggested in the context of a series of research activities based on Kuhn's work rather than as a fundamental mechanism for examining the validity of the analogy between Kuhn's models and CBSD itself.

I conclude that the easiest way to provide support for the Kuhn-based model of CBSD is by looking for the DM and its effects. This chapter therefore describes a series of investigations designed to articulate the DM by looking for elements which might be included in such a belief system. The elements of the DM of CBSD found during these investigations, and others identified during reading for the work reported in this thesis, are summarised in Appendix 2.

5.2.1.3 Dangers in Looking for Elements

There are both theoretical and practical issues arising from a search for elements of the DM of a discipline.

Looking initially at Kuhn's own beliefs, he "...insists that there is more to a paradigm than can be explicitly laid down in the form of explicit rules and directions" (Chalmers, 1992, p 93).

Chalmers' comment implies that I cannot claim to be able to identify all of the elements of the DM of CBSD, since some of these may remain elusive and impossible to specify. I certainty cannot claim completeness in the early stages presented here of what might be a long search. Whether the DM can be completely identified is a question which can only be answered by more detailed and systematic research, building on the initial exposition in this thesis. However, I do not believe that Chalmers' statement invalidates my approach.

There are also practical dangers in trawling though the work of others looking for their underlying belief and models. These include:

- reading too much into statements which are not intended to convey what is supposed, particularly in cases where I may have expected to see certain results and may therefore have 'found' them;
- double filtering on the reported results of others' work – this is covered in more detail below when examining the most likely problem area, that of action research; and
- drawing general conclusions in my own work from specific examples, such as locating an element from a source, then looking for it in *that source* again to 'show' that it does exist; this also make it difficult to draw conclusions as

to the applicability of elements to a certain aspect of CBSD based on only one example.

I have attempted to minimise these risks in my informal investigative mechanisms, by examining a variety of types of sources and, in the case of the first investigation, by confirming the findings of a trawl through one set of sources in another set of similar sources.

5.2.2 The Mechanisms Employed

The work reported here comprised initial investigations, designed to explore a situation before it was possible to design theory-based, formally specified experiments. As a result, I have adopted an informal approach to the practical work.

I have employed two mechanisms in the investigative work. These were trawling through published literature, *viz.* textbooks and published reports, and interviewing practitioners.

Trawling through textbooks was adopted in order to identify those elements of a DM which are held by those who develop and teach CBSD methods and tools, and which might influence practitioners as they are taught from these sources. As noted above, Kuhn regards textbooks as a source of existing implicit beliefs for those joining a discipline. Textbooks were therefore chosen for investigation since they encapsulate the *theory* of the discipline of CBSD.

Interviewing practicing computer-based systems developers allowed DM elements found in the textbook trawls to be checked, and allowed for further elements, found in the *practice* of CBSD but not well documented in the textbooks, to be identified.

In the minor investigations, the explicit lessons learned from an action research programme into the usefulness of a particular CBSD framework, Multiview (Avison and Wood-Harper, 1993), and a report on a well-documented failed project (LAS, 1993), were examined using the trawling mechanism.

Since the investigative mechanisms adopted were informal, their results can, in the main, only be treated informally. In consequence, I have performed no

quantitative analysis on the results, with the exception of a consideration of the implications of finding a DM element in all of the sources used in the second phase of the book trawl.[29]

5.2.3 Looking for Quality

5.2.3.1 Introduction

In the major investigations, the trawls through textbooks and interviews with practitioners, I have found it useful to look at one aspect of CBSD in particular. The aspect selected was that of the 'quality' of the processes or products of CBSD.

The reason for selecting one attribute as a starting point for searching the sources is purely practical. This mechanism allows me to use a series of textbooks to gain some breadth and comparability in the task of examining the current state of CBSD, without undertaking the large amount of work involved in looking through the complete texts of those books.

5.2.3.2 Why Quality Was Selected

I selected an examination of the term 'quality' as a position from which to identify underlying assumptions in the sources because:

- it is an aspect of CBSD which the sources have to address in describing or performing CBSD, and which therefore will be present in each, either implicitly or explicitly;
- as will be shown, it reveals relatively unambiguous information about the thinking behind the determination of the criteria for its evaluation; and
- of the relative richness of the beliefs revealed on the part of the authors and interviewees.

The selection of 'quality' as a starting point for the investigations implies a number of hypotheses. These are that:

- in at least some of their work software teachers and developers aspire to teaching or applying some criteria in order to determine whether the level

---

29 For a description of this analysis, see Chapter 6, below.

of software 'quality' (however defined) is sufficient for the circumstances. Therefore examining this aspect and seeing how it is defined will reveal some of the underlying beliefs of those concerned;

- having defined it in some way, people attempt to achieve 'quality' in their work. If the definitions of this factor can be shown to differ in any material respects between people, then support is obtained for the Kuhn-based model of CBSD theory, as being based on a number of schools, each of which may, *inter alia*, define 'quality' differently; and
- the explicit or implicit inclusion of a belief or model in an author's or interview subject's definition of 'quality' in respect of software and/or its development is sufficient for the purposes described here to make its identification as an element of his or her DM useful and valid.

Additionally, if differences were found between the belief systems underlying definitions of software quality which are based on the techniques, notations or methods used, then it could be inferred that the process which is adopted for developing software can affect the products of that process. To translate that into the terms of the theory presented in this thesis, the DM underlying a particular CBSD process affects the products of that development.

5.2.3.3 Defining Quality – or Not

I have not attempted in this exercise to provide my own definition of 'quality' in CBSD processes or products. Any attempt on my part to define or pre-judge a definition of 'quality' before conducting the investigations would inevitably have resulted in my missing aspects of others' definitions of 'quality' which I had excluded from mine. In this instance, the lack of an initial hypothesis as to what constitutes 'quality' allowed a broader examination of the sources used.

In view of the objectives of the investigations, relating to the DM and its elements, when examining others' definitions of 'quality' I did not seek to discriminate between different aspects and usages of the term 'quality' in respect of CBSD processes and products, except insofar as these differences revealed different underlying beliefs. The results presented below and in Appendix 2 are to be read in the context of a search for the shared commitments among the CBSD community. They do not distinguish at a high level between groups of beliefs and models specifically related to such aspects or contexts of consideration as:

- the internal design, structure and characteristics of a software system;
- the process of developing software;
- the fitness for its purpose of a software system, and/or how to determine this; or
- whether or how the satisfaction of the users of a system is to be determined and/or maximised,

although some of the beliefs identified, such as those based on software process or product standards, relate explicitly to some contexts only.[30]

5.3 Trawling Through Textbooks

5.3.1 Introduction

This section describes the first investigation performed to examine the validity of the Kuhn-based model of CBSD, *viz.* a trawl through a number of textbooks looking for underlying elements of the DM of CBSD. The results of the first phase of this investigation have been presented at a conference (Winder and Wernick, 1994)

5.3.2 Objectives of the Investigation

As the book trawling was the first investigative work performed on the Kuhn-based model of CBSD, the objectives of the work were set relatively broadly.

In this trawl, I therefore sought general confirmation of the existence of a DM for CBSD, by looking for beliefs and/or models common to the sources examined. To summarise, the rationale for the work was to see whether an identifiable DM exists, i.e. the first of the broad objectives listed above. This was attempted by looking for examples to support the second objective, *viz.* identifiable elements of that DM.

---

30 The subdivision of the elements listed in Appendix 2 under a set of headings is a result of my *post facto* analysis of the results of the investigations, and does not derive directly from any of the investigative work.

5.3.3 General Comments on The Procedure

The mechanism of this trawling work on textbooks was structured to be as sound as possible given its informality. It was deliberately designed after the form of an experiment, using a first trawl to develop a 'theory' which could then be tested in a second trawl.

The work was therefore structured as follows:

- conduct a first trawl through a number of textbooks, looking for beliefs and models underlying aspects of 'quality' as defined by the authors;
- present the results in the form of a list of elements of the DMs represented by the books; and
- perform a second trawl, comparing the compiled set of beliefs with another set of textbooks, to determine the extent to which the structure of beliefs underlying quality in the new books matched that found for those already examined in the earlier trawl.

5.3.4 The Detailed Procedures

5.3.4.1 Phase 1: The Initial Trawl and its Results

The sources used for the first phase of this work – searching for elements – were:

- Birrell and Ould *A Practical Handbook for Software Development* (1985), a UK-written book designed to compare methods for the edification of software practitioners based on a management-oriented framework defined by the authors;
- Sommerville *Software Engineering* (1992), the fourth edition of a general textbook written in the UK, earlier editions of which have been oriented towards the Ada community;
- Downs *et al. SSADM – Design and Context* (1992), a UK textbook describing a single method in depth and therefore committed to that method; and
- Licker *Fundamentals of Systems Analysis with Application Design* (1987), a Canadian-written, USA-published textbook designed for the "non-technologist".

The summarised results of the trawl's first phase are presented here in the form of a table. The order in which the beliefs and models identified are presented here is alphabetical, and is not intended to describe any possible relationships between the elements.

Element	*Birr*	*Somm*	*Downs*	*Licker*	*Total*
(Possibly) a low assumption of reliance on computer-based support.	Y	-	-	-	1
(Possibly) consideration of the external aesthetics of computer systems.	-	-	-	Y	1
A compromise must be made between conflicting priorities during the systems development process.	-	Y	-	Y	2
A particular computational model should be adopted for the system design (procedural/data for Birrell and Ould, procedural/object for Sommerville, data for Downs et al., procedural for Licker).	Y	Y	Y	Y	4
A quality-aware process will result in a higher quality product.	-	-	Y	-	1
A sequential rather than parallel mode of operation within a module.	-	Y	-	-	1
Abstraction as a feature of a design is a good thing.	-	Y	-	Y	2
Based on the example given, there are 'good' values for the minimum and maximum numbers of elements in a diagram.	-	-	-	Y	1
Computer systems should be built up from components rather than being monolithic.	-	Y	-	Y	2
Computer systems should be designed to be portable.	Y	-	-	Y	2
Controlling the software process produces a better product.	-	Y	Y	-	2
Controlling the software process produces benefits which outweigh the costs.	-	-	Y	-	1

Element	*Birr*	*Somm*	*Downs*	*Licker*	*Total*
Decomposition and composition do not alter in a negative fashion the nature of the thing broken down or built up to a significant extent (or the benefits of such decomposition or composition outweigh the disadvantages).	-	Y	-	Y	2
Design for modifiability, (which might be considered an aspect of maintainability).	Y	-	-	-	1
Design for reuse.	Y	-	-	-	1
Disagreement with the view that people interacting with a system can be treated as objects, and reduced to roles.	-	-	-	Y	1
Efficiency.	Y	-	-	-	1
Elegance.	Y	Y	-	-	2
Following fashion, in this case the stylistic lead given by safety-critical systems.	Y	-	-	-	1
Independence of specific aspects of the development, such as the implementation language, is NOT important when designing notations, etc.	-	Y	-	-	1
Interactive computer systems should be self-documenting on-screen.	-	-	-	Y	1
Self-modifying systems are specifically considered to be bad things.	Y	-	-	-	1
'Software engineering' is in some respects like other 'engineering' disciplines.	Y	-	-	-	1
Software process standards are of importance in maintaining software quality.	-	Y	-	-	1
Software product standards are of importance in maintaining software quality.	-	Y	-	-	1
Software quality can be assessed by quantitative measurements.	-	Y	-	-	1
Solutions can be better generated by a process of breaking the complete solution into smaller bits.	Y	Y	-	Y	3
Systems should be designed so as to minimise the effects of bugs.	Y	Y	-	Y	3

Element	*Birr*	*Somm*	*Downs*	*Licker*	*Total*
Systems should be designed to be maintainable.	Y	Y	-	Y	3
The costs and timescales of a computer system development can be estimated in advance with a reasonable degree of accuracy.	-	Y	-	-	1
The software development process is capable of being managed.	-	Y	Y	-	2

5.3.4.2 Phase 2: The Second Trawl and its Results

In the second trawl, the beliefs and models found during the first trawl were searched for in another group of books.

The sources used for the second phase were selected to match the first group in style, and in my perception of the authors' objectives. I therefore used:

- van Vliet *Software Engineering: Principles and Practice* (1993), a UK edition of a Dutch undergraduate textbook selected as an analogue to Sommerville's book;
- Jackson *System Development* (1983), a UK single-method text describing Jackson's 'JSD' method, selected to counterpoint Downs *et al.* on SSADM;[31]
- Schach *Software Engineering* (1993), a USA college-level text, now in its second edition, designed for use in project-based 'Software Engineering' courses for "graduate students as well as upperclass undergraduates", which I have used as a parallel to Licker's work;
- *Object Development Methods* (Carmichael, 1994), an edited volume of contributed papers for "practitioners and academics", "applicable for those who are presently choosing a method or for those who have already chosen a method and wish to become familiarized with other methodologies" (Carmichael, 1994, paperback edition, back cover notes) – the book includes descriptions, of lengths varying from 9 to 35 pages, of a number of object-oriented CBSD methods, of which Booch's (1994) description of his own object-oriented method is selected for examination here, due to its

31 It should be noted that Jackson's (1983) book used here is the original text on the JSD method, intended to be both a descriptive and a pedagogic work.

well-documented advice for the method's users, particularly as set out in a "Summary of Recommended Practices" (Booch, 1994, pp 164–165)

The search was made in the texts of these sources for the elements previously found in the following order:

- in any explicit definitions of 'quality'[32] in the sources, as noted by sections devoted to the subject;
- explicit mentions in other parts of the book, indicated by index entries under 'quality' and associated headings such as 'quality control', 'quality assurance', etc.; and
- more speculative or implicit identifications, based on my perceptions of the underlying assumptions made by authors in ascribing 'quality' to aspects of the CBSD process and products, such as criteria for a 'good' design (e.g. considerations of cohesion or coupling).

Additionally, note was taken of any of the elements listed which are explicitly denied by the second group of sources, and of any elements for which evidence of validity could be found without my being able to determine the authors' opinions.

No attempt was made to find all of the evidence in respect of a particular element, so the evidence given here is not necessarily exhaustive in a source. Searching terminated either when enough evidence was found to support a conclusion that an element was believed in or not, or that it was impossible to tell the authors' opinions from their written works examined here. Also, since the second group of textbooks were examined for evidence related to the list of beliefs found in the first phase of the trawl and not explicitly for other beliefs implicit or explicit in them, the evidence obtained from this second trawl is not comprehensive outside that list.

The results obtained in the second phase of the investigation are summarised here, in a table which relates them to the results of the first investigation.

The entries in the column for each source have the following meanings:

---

32 As before, the definition of the term 'quality' employed here is that of the sources' authors; no analysis of this term was made in the investigation.

Q	=	agreed under 'quality' (q = weak agreement)
A	=	agreed other than under 'quality' (a = weak agreement)[33]
D	=	disagreed (d = weak disagreement)
?	=	some evidence, but position unclear
-	=	subject not covered in the text

Element	*vanV*	*Jack*	*Schac*	*Booch*	*No. agree*	*No. find*
(Possibly) a low assumption of reliance on computer-based support.	D	D	D	d	0	4
(Possibly) consideration of the external aesthetics of computer systems.	a	-	D	-	0 + 1	2
A compromise must be made between conflicting priorities during the systems development process.	Q	A	Q	A	4	4
A particular computational model should be adopted for the system design.	Q	A	Q	A	4	4
A quality-aware process will result in a higher quality product.	Q	A	Q	A	4	4
A sequential rather than parallel mode of operation within a module.	-	A	?	-	1	2
Abstraction as a feature of a design is a good thing.	Q	?	Q	A	3	4
There are 'good' values for the minimum and maximum numbers of elements in a diagram.	-	-	a	-	0 + 1	1
Computer systems should be built up from components rather than being monolithic.	Q	A	A	A	4	4
Computer systems should be designed to be portable.	Q	A	Q	-	3	3
Controlling the software process produces a better product.	Q	A	Q	D	3	4
Controlling the software process produces benefits which outweigh the costs.	Q	A	A	A	4	4

33 Neither Jackson's (1983) nor Booch's (1994) description covers 'quality' as a separate issue, and therefore all references found in these sources are denoted by 'A'.

Element	*vanV*	*Jack*	*Schac*	*Booch*	*No. agree*	*No. find*
Decomposition and composition do not alter in a negative fashion the nature of the thing broken down or built up to a significant extent (or the benefits of such decomposition or composition outweigh the disadvantages).	Q	A	A	A	4	4
Design for modifiability.	Q	A	Q	A	4	4
Design for reuse.	Q	-	Q	A	3	3
Disagreement with the view that people interacting with a system can be treated as objects, and reduced to roles.	q	?	a	A	1 + 2	4
Efficiency.	Q	?	-	-	1	2
Elegance.	q	-	Q	A	2 + 1	3
Following fashion.	Q	?	Q	A	3	4
Independence of specific aspects of the development, such as the implementation language, is NOT important when designing notations, etc.	?	?	-	A	1	3
Interactive computer systems should be self-documenting on-screen.	Q	-	A	-	2	2
Self-modifying systems are specifically considered to be bad things.	-	-	-	-	0	0
'Software engineering' is in some respects like other 'engineering' disciplines.	A	-	A	-	2	2
Software process standards are of importance in maintaining software quality.	Q	-	Q	a	2 + 1	3
Software product standards are of importance in maintaining software quality.	Q	-	Q	a	2 + 1	3
Software quality can be assessed by quantitative measurements.	Q	-	Q	A	3	3
Solutions can be better generated by a process of breaking the complete solution into smaller bits.	Q	D	A	?	2	4
Systems should be designed so as to minimise the effects of bugs.	Q	-	Q	A	3	3
Systems should be designed to be maintainable.	Q	A	Q	A	4	4

Element	*vanV*	*Jack*	*Schac*	*Booch*	*No. agree*	*No. find*
The costs and timescales of a computer system development can be estimated in advance with a reasonable degree of accuracy.	a	?	D	-	0 + 1	3
The software development process is capable of being managed.	Q	A	Q	A	4	4

5.3.5 Lessons Learned from the Trawls

5.3.5.1 General Conclusions

I summarise here the general conclusions which I have drawn from the work performed on textbooks:

- The Kuhn-based model of CBSD was not disproved by the trawl. In Popperian terms, this is the strongest statement which can be made about the validity of the model, i.e. that it has not yet been falsified as a theory.
- Trawling for elements of the DM works; the mechanism adopted to look for beliefs and models underlying CBSD theory results in the identification of such things.
- Some support for the Kuhn-based model of CBSD, although not for a single unified discipline, in that:
  - some elements identified as being in the DM of CBSD are possibly common to the discipline, as evidenced by their being found in all sources for the second phase;[34] and
  - some elements seem to divide the authors into sub-groups, based on their adoption or rejection of them.

  Kuhn's theory would lead us to believe from this conclusion that either the discipline of CBSD is divided into sub-disciplines based only on the application area covered, or it is formed of differing schools of thought, as might be characteristic of a pre-science discipline. By looking at the books

---

34 In the list of elements in Appendix 2, I have included all of the elements identified in the first trawl, for which support was found in all of the sources used in the second trawl, as common to all computer-based systems developers, with the exception of three which I regard as dividing CBSD into schools. The rationale for making these exceptions is given in Chapter 6 below.

examined, which suggest that their mechanisms are general to more than one area of CBSD, the former is not correct.

Therefore the conclusion can be drawn that the theorists of the discipline of CBSD are divided into schools of thought, which is what would be expected from the pre-science model of CBSD theory.

- Different writers ascribe different meanings to the same terms. For instance, Birrell and Ould's (1985) definition of the types of cohesion, based on Myers (1975), differs from that of Sommerville (1992), which builds on Constantine and Yourdon (1979), even though most of the types differ only in name rather than in definition.

  This is also to be expected from the Kuhnian pre-science school-based model of CBSD theory.

5.3.5.2 Concerning 'Quality' in Computer-based Systems Development

What lessons can we learn specifically about the nature of 'quality' in CBSD from the trawling work described above?

That opinions differ among the sources used for the trawling work as to the relative importance of the different contexts for the examination of 'quality' of itself lends support to the Kuhn-based model of CBSD, in particular the proposal of an analogy with the pre-science school stage of a discipline's development, based either on differing beliefs or the acceptance of different values for the same belief. However, some evidence has arisen supporting the idea of the existence of a core set of beliefs common to all computer-based systems developers who adhere to the theories presented in the textbooks.

As an interesting example of, and slight evidence for, one particular view as to the proper context of 'quality' and for the existence of more than one such context, I note an entry in the index to Schach's book used in the second trawl: "quality (*see* software quality)" (Schach, 1993, p 576). Does this indicate that Schach (or his indexer) only sees 'quality' in terms of the products of CBSD? Were there only one context for the consideration of 'quality' in the discipline of CBSD, such an entry might be redundant since confusion could not arise in the minds of Schach's readers as to the context of the 'quality' being described. As it is, I feel that the index entry which I have quoted opens the question as to whether other contexts for the consideration of 'quality' exist in CBSD. An example of another possible

context is that of the 'quality' of a *process* of developing software, and questions concerning process quality were included in the questionnaire used to structure the interviews performed as part of the work presented here (see Appendix 1).

Perhaps most crucially, it appears from the results obtained that people judge 'quality' in assessing models in terms of the attributes of those models, and therefore any features missed by the tools used to develop those models (such as those underlying computational models) will be missed in assessing quality; this indicates a mechanism whereby the underlying DM adopted with a systems analysis, design or implementation mechanism might affect the results obtained by the use of that mechanism.

It must be observed that all of the conclusions drawn above are only supported so far in respect of the *theory* of CBSD. By looking at textbooks, we can observe the nature of the discipline as it is likely to be taught. A further investigation is required to see whether this situation is actually reflected in the practice of developing computer-based systems, and this is described below.

5.4 Talking to Practitioners

5.4.1 Introduction

In a second informal investigation into the Kuhn-based model of CBSD, I interviewed eight people who develop, or have developed, computer-based systems. These interviewees were selected to reflect the use of a variety of CBSD mechanisms in a number of different system and organisational contexts.

5.4.2 Objectives of the Investigation

Having determined that the theoretical side of CBSD at least seemed to be based on a number of beliefs and models, some of which appeared to be sufficiently well defined to facilitate their identification in CBSD practice, I decided to look for these beliefs and models in the attitudes of CBSD practitioners. I was also able to build on the existing investigative work by looking for additional elements to enlarge the DM.

In this investigation, I was able to take advantage of the experience gained in, and the results of, the textbook trawling work reported above, to formalise the objectives to a greater extent than had been possible in the first investigation.

The resulting formalised null hypotheses were that:

- an identifiable DM does not exist in CBSD; and
- the individual elements of the DM identified during the book trawl cannot be identified in the practice of the discipline.

Stated less formally, the objectives of this work were to:

- demonstrate that at least some of the DM elements identified for the *theoretical* side of CBSD are also to be found in the *practice* of such work;
- extend the number of such beliefs and models identified; and
- provide evidence to support the existence of a DM in the practice of CBSD.

5.4.3 Selecting the Participants

The interviewees were selected to cover a range of CBSD activities and contexts, and of varying degrees of use of formalised CBSD methods, tools and techniques. These backgrounds are summarised below. Since, due to the limited resources available, it was possible to interview only a small number of subjects, it was not possible to cover more than a subset of all current development environments and circumstances. The interviewees were also selected from those easily available via personal contacts, to render the process and atmosphere less formal and make the subjects more likely to speak freely.

The interview subjects therefore cannot be assumed to be typical of the full range of CBSD practice and contexts. For instance, no interviews were related to the activities of maintaining existing software applications or porting software to new target machines. However, the participants were selected to reflect some diversity in CBSD, with particular reference being made to the following criteria in selecting both the interviewees and the examples of their work discussed in detail:

- whether the subject is a 'convert' to CBSD work from another career or was initially trained in CBSD and has had no other type of employment;
- the size and the type (academic or industrial/commercial) of organisation for which the developer performs or performed his or her CBSD work;

- the size (from one person to a large development team) of the development team;
- whether, if in a large development group, the developer works/worked as an individual, developing his or her software without depending on the availability of the results of current work of others on the same project (as might be typical of an academic research environment), or in a team development environment in which the developer shares or shared either the phases of development, or the work within a specific phase;
- whether the particular example of CBSD work used as a focus for the interview was towards an end user or an embedded system, and its size (based on the number of lines of source code in the implementation);
- the stages of the software life cycle from initial problem to replacement dealt with by the participant, directly and/or as manager;
- whether or not a formalised, predefined CBSD method (process model, techniques, documentation) is/was used;
- whether advanced CBSD tools (such as CASE and workbench tools) are/were used to support CBSD; and
- whether the developer can/could set his or her own standards, or works for an organisation which sets its own standards beyond the developer's immediate control.

Whilst the small number of subjects has not permitted the correlation of the results obtained with each of these criteria, the breadth of coverage enables a measure of confidence to be gained that the results might be typical of the generality of CBSD practice, more so than would be the case were the subjects to have been drawn from the same background and type of work.

Additionally, each subject was asked at the start of the interview whether he or she was aware of the purpose of the interview, in order to determine whether their answers might be biased towards what they believed to be my desired objectives in the interview. None was so aware.

5.4.4 Backgrounds of the Subjects

In order to preserve the anonymity of the interviewees, they are denoted by single letters from B to K. The sequence of letters is not continuous in this range, A, D

and I being omitted for the following reasons. The interview with subject A was reoriented towards a discussion of general engineering design after it emerged that his experience was not relevant to this work, being related purely to electronic hardware design rather than CBSD. 'Subject D' is an electronics engineer who was not an interview subject for this phase of the investigative work, but whose opinions I solicited as part of an examination of 'engineering' in general. The letter 'I' was not used to avoid confusion with the personal pronoun.

Brief backgrounds of the subjects interviewed for the work presented here are given in the following table. The work contexts are based on the specific work examined in the interviews, rather than on any more recent work.

Subject	Convert to Systems Dev.?	Organisation Size and Type[35]	Individual/Group Environment	System Type/Size[36]
B	No	Large commercial	Individual	Embedded/med.
C	Yes[37]	Large academic	Group	Embedded/large
E	Yes[38]	Large commercial	Group	End user/large
F	Yes[39]	Small commercial	Individual	End user/large
G	Yes[40]	Large commercial	Group	End user/medium
H	No	Large commercial	Group	End user/large
J	No[41]	Large commercial	Group	End user/large
K	Yes[42]	Large commercial	Group[43]	End user/medium

35 For these purposes, the organisation is defined as 'small' if it had less than five employees in total (including non-systems development staff), 'medium' if five to 100 employees, and 'large' if more than 100 employees. Also, 'commercial' covers all types of commercial and/or industrial environments.

36 For these purposes, the system is defined as 'small' if it comprised less than 1000 lines of code, 'medium' if between 1,000 and 10,000 lines, and 'large' if more than 10,000 lines.

37 Former secondary school mathematics teacher.

38 Trained as a mechanical engineer, converted to CBSD via O & M

39 Initially in advertising and marketing, trained in programming, then trainee systems analyst.

40 Trained and worked as an economist, became an end user, supporting one specific system including specifying changes, O & M and user support, finally to development.

41 First degree in electronic engineering with an emphasis towards computing, an MSc in business systems analysis and design, then into commercial computing.

42 First degree in chemistry and geology with a computing-related project, research into organic chemistry but interest in computing resulted in a move into commercial computing.

43 As a consultant, performing the difficult work for groups of developers in the same organisation.

Subject	Life Cycle Stages	Method Used?	CASE etc. Used?	Standards Set By ...
B	All	No	No	Developer
C	All but coding	No	No	Developers
E	Analysis and design	Informal	No	Informal
F	All	No	No[44]	Developer
G	All but coding	Yes	No	Method, developer, organisation
H	Analysis, design, implementation	Yes	Yes	method, organisation
J	Analysis, design, implementation	Yes	Yes	method, organisation
K	Analysis, design, implementation	Yes	Yes	method, organisation

It should be noted that the work of the participants which was selected for examination in the interviews, and is reflected in the table above, was that which matched the desired spread of criteria, rather than being necessarily their most recent or current CBSD activities. For most interviewees, the work considered is either their most recent or current CBSD work, or a fairly recent example.

Since the results are based on discussions with individuals reviewing their own past or current activities, the conclusions of the interviews are to some extent based on the subjects' self-assessment of their work. I recognise that this approach risks capturing the memories of the interviewees as coloured by more recent experience, but it did allow a wider range of CBSD work to be covered than might otherwise have been the case. The use of non-intrusive observation studies into computer-based systems developers at work to overcome subjects' subjective input is suggested under 'future work' in the conclusion to this thesis.

5.4.5 Detailed Procedures

The approach taken to this part of the investigative work was that of a trawl rather than that of an experiment. Although I was hoping to identify some existing beliefs and models of CBSD theory in its practice, I conducted the interviews on the basis that I was looking to see whether I would find elements of a belief system underlying the subjects' practice, rather than checking off predefined elements of

[44] Flowcharting program only used, no method-specific support tools.

such a belief system. The objective in each interview was therefore to lead the discussion towards the elicitation of (initially unknown for CBSD practice) belief system elements from the subjects.

Since I was unable to determine in advance the results which I was to achieve, I used semi-structured interviews, based on a questionnaire[45] which was used as an *aide memoire* to guide the flow of the interviews rather than as a list of direct questions requiring direct answers. This approach reduced the risk of leading the interview subjects towards giving answers which might have appeared to have been expected by me or seen by the subject as being the 'right' answers in the context of the discipline. The adoption of a more direct, structured interview approach might have predicated the subjects or the interviews themselves towards particular real or imagined elements. Alternatively, it might have hidden other elements, not previously recognised and therefore not looked for in the interview. It should be noted that even the semi-structured approach runs the risk of such hidden input by the interviewer into the investigative process prejudicing the results. However I felt that the semi-formal mechanism which I used minimised this risk.

The conduct of the interviews was based on indirect questioning, designed to elicit the thinking behind explicit beliefs rather than those explicit beliefs themselves. The most important aspect of this related to the interviewees' beliefs concerning the nature of software quality. They were explicitly asked what comprises 'quality' in a software product and in a software development process, and for their definitions and opinions of 'Software Engineering'. Their answers were analysed for both explicit beliefs and models, and for any implicit aspects which their responses revealed.

Notes were taken at and in some cases immediately after the interviews, and a tape recording made of each discussion. The recordings were transcribed into a word processor file, and these formed the basis of the subsequent analysis.

5.4.6 Analysing the Transcripts

As stated previously, the interview-based work was performed as an informal investigation rather than as a scientific experiment. This approach extended from the interview procedures to the interpretation of the results.

---

45 The questionnaire used to structure the interviews, reformatted for consistency with the rest of this text, is reproduced as Appendix 1 to this thesis.

For each transcript, having annotated it for nuances such as the tone of voice used by the subject to bring out feelings not conveyed by the unadorned words, I then read it looking for evidence of the existence of underlying beliefs and models. I looked in particular for those beliefs which the questionnaire used to guide the interview was designed to elicit, i.e. quality-related beliefs and the subject's view of 'Software Engineering'. Additionally, any beliefs or models identified in the attitudes and activities of the organisations for which, and the colleagues with whom, the subject worked were noted.

In performing this analysis, I looked particularly for beliefs which were not unique to the individual subject or organisation, but which could be generalised at least to part of the CBSD community. These decisions were based on my personal experience and knowledge of CBSD and the current state of its practice, and are therefore open to re-interpretation. The DM elements found and listed in Appendix 2 reflect my opinion of the subjects' opinions, rather than being a 'true' analysis of the transcripts.

Since the interview work was performed at a stage of the research where no well-defined DM existed to use as a reference point, formerly unrecognised elements were added to the existing DM as they were identified in each transcript. When an element of the DM was found for the first time during the examination of a particular transcript, earlier transcripts were not re-examined to determine whether it was implicitly present in those earlier interviews. The process was one of accretion of new information rather than one of checking transcripts for previously identified elements.

5.4.7 Summarising the Results

In terms of the informal objectives previously noted, the results of the interviews can be presented as follows:

- of the 31 of the DM elements found during the textbook trawling work, 20 were found in the practice of CBSD as reflected in the comments (whether positive or negative) of the interviewees;
- a number (220) of additional elements of the DM of CBSD as practised were identified; and

- overall, a structure of beliefs and models can be identified in the practice of CBSD which is similar to that which has been identified for CBSD theory.

In terms of the formalised null hypotheses:

- evidence has been accumulated to suggest that a belief system underlies the work of the CBSD practitioner, a belief system which might be identified with a Kuhnian DM; and
- the assertion that the DM elements found for CBSD theory cannot be identified in its practice is falsified in most instances.

The detailed results of the investigations, in terms of the elements identified in the DM of CBSD, are presented in Appendix 2. They, and the structure within which I have grouped them, are discussed in the next chapter, in the context of a more detailed description of the Kuhn-based theory of CBSD. Both the elements and the headings (other than Kuhn's division of the DM into symbolic generalisations, metaphysical and heuristic beliefs and models, values and exemplars) are expressed in my words, rather than being drawn from any one of the sources used to find them. In some cases, this has resulted in elements being reworded from those listed as the results of the book trawl earlier in this chapter. For example, the element described previously in this Chapter as 'A particular computational model should be adopted for the system design' is reflected in Appendix 2 in more general terms as 'All software design is with respect to a particular, identifiable computational model'.

The results for each element in the list are presented informally, in terms of the number of interviewees agreeing and/or disagreeing. I have made no attempt either to perform any statistical analysis on the results which I have obtained, or to correlate these results with the interviewees' backgrounds, since the basis of the investigative method is in my opinion not sufficiently rigourous to support such an action and in view of the small number of subjects in this work.

The appearance of a number in the 'interview' column in the table in Appendix 2 also does not imply that the interviewee believes in or supports that entry him- or herself. It is intended rather to demonstrate that such a belief or model is either actually prevalent, or by the actions of the people involved appears to be prevalent, among some or all of the CBSD practice community. Evidence to

support this hypothesis might be found in the interview transcript in comments referring to the interviewee, the methods or tools used, his or her employers or clients, or his or her view of the discipline of CBSD in general.

An observation which can be made on the table in Appendix 2 is that in a small number of cases, interview subjects seem simultaneously to agree and to disagree with particular beliefs. These are indicated in footnotes to the table. In some cases, this reflects evidence provided by the subject concerning more than one of the possible sources of elements, commonly conflicts between his or her personal viewpoint and that of his or her employer. However, in other cases both supportive and disagreeing evidence is found as to the subject's own beliefs. For instance, subject K provides weak support for and against the proposition that 'Complete freedom from faults is a valid objective of computer-based systems development'. In support of this proposition, he said during the interview that a definition of quality in a software product includes "it doesn't have faults in it" (transcript, p 17); against this, he had earlier stated that "the thing is that in the commercial world, you in a way, you measure success by whether the user pays the bill at the end of it", (implicitly) not by complete freedom from product defects (p 11).

This apparent internal inconsistency in a subject's beliefs may indicate a weakness in the mechanisms employed in finding elements, either in the conduct of the interviews, or in the analysis of the results, especially when weak implicit evidence is looked for in a subject's words. Alternatively, it might indicate that the subject is actually unsure of his or her views on this aspect of CBSD, or that he or she might think of it differently in different contexts. For instance, in the example quoted above, the subject's comment in support of fault-free software is in the context of a discussion of definitions of software quality and might have been related to the user's perception of the number of faults in a software product, whereas the statement disagreeing with this idea was made whilst talking about the definition of success in a commercial CBSD project. Different contexts might provide different results, and this should be considered in future work designed to identify specific elements of the DM.

In the context of an investigation intended mainly to determine whether sufficient common ground could be found between computer-based systems developers to support the general theories advanced in this thesis, I believe that any weakness indicated in the investigative mechanisms is not a significant problem with the

mechanisms themselves. However, this issue would have to be addressed were more formalised interviews to be used in later research for the identification of specific elements.

Finally, the transcripts revealed that there were disagreements in some cases between members of a CBSD team as to how to prioritise the goals of a CBSD project. An example of this is a difference of opinion between subject E (transcript, p 20) and his manager concerning the relative importance of good analysis and thorough testing (the subject) and having the customer pay the bill (his manager).

5.4.8 Commentary on the Results

The non-experimental basis of the investigation is reflected in the use of, and claims made for, the results.

I suggest that the results I have obtained are informally generalisable to the extent that another worker using the same investigative mechanism on a closely equivalent group of subjects would obtain broadly the same results, and that the results which I have obtained are broadly true for those parts of the CBSD community which I have covered.[46]

As previously stated, I have not tried to correlate the elements found with the backgrounds of the subjects. The reason for selecting a wide range of subjects was to make the scope of the work as wide as possible rather than to allow such correlations to be made. In view of the small number of subjects interviewed, I feel that any correlations of belief systems with backgrounds, development contexts etc. are premature, and cannot be supported statistically. Such work could be performed using closed questionnaires and tightly selected samples of developers, and I suggest under 'future work' below that such work should be performed.

I have also not looked to the possible sources (discipline, method, technique, etc.) within CBSD of the elements disclosed in this work, since the informal mechanisms were not felt sufficiently sensitive to produce results for this other than by direct questioning of the subjects, which as noted previously has its own

---

46 For example, the investigation has not covered aspects of software 'quality' as defined by adherents of formal methods.

pitfalls. In presenting the results I have therefore not distinguished between elements in the DM due to the subject, and those highlighted in the subject's employer and/or the method/tool developers etc.

I am therefore unable to distinguish the source of any particular element and how it is operationalised in CBSD theory and/or practice. Investigations into this area form a useful addition to future work in this research programme.

5.5 The Minor Investigations

5.5.1 Introduction

In addition to the major investigations described above, two smaller investigations were conducted, as a means of widening the scope of the evidence for the Kuhn-based model and for the existence of the DM. In addition, these acted as an additional trawl, providing further elements for the DM.

5.5.2 Trawling a Report of Action Research

5.5.2.1 Introduction

In this section I describe a trawl for elements of the DM of CBSD in the reported results of a practically-based action research programme. The method which I examined in this way was Multiview, a multi-stage method, described in a book (Avison and Wood-Harper, 1993) and in various published papers. The book alone was used as the source of the analysis described here.

5.5.2.2 The Source and How it was Used

According to the explanatory material on the rear cover of the paperback edition of the book which describes Multiview (Avison and Wood-Harper, 1993), the book is "suitable for data processing, systems analysis and information processing courses at both Undergraduate and Master's levels, and researchers looking at the question of information systems development. It will also be of interest to practitioners and managers." The book can therefore be considered to be primarily a pedagogical textbook, summarising the outcome of research contained in published papers.

Within the book, the main source which I used to extract DM elements is Chapter 19; 'Theory and Practice'. This chapter is further divided into 'Lessons from the Field Work' and 'Conclusions from this Experience'. The chapter is intended to "draw conclusions on the way Multiview worked for us in the cases that we have described..." (*ibid.*, p 265), and includes "reflections on the experience on developing Multiview as well as attempting to draw lessons from the experience of using it." (*ibid.*, p 265).

5.5.2.3 Countering Double Filtering

Throughout this part of my work, I have been conscious of the danger of 'double filtering', in which the raw information obtained during the action research studies is processed by the experimenters to produce the results ready for publication, and a second investigator then uses these published results without reference to the fact that they are not raw data, but have already been processed.

In order to minimise this effect, I have, where possible, started by considering as sources of elements of the DM the explicit conclusions drawn by the original investigators, and examined these for evidence of underlying beliefs. Their conclusions are presented as the results of real world or modelled investigations. I have looked at these conclusions and asked whether other beliefs have coloured the results which they have obtained and the conclusions which they have drawn.

5.5.2.4 Results of the Investigative Work

The results of the examination of this source are reflected in the table of elements in Appendix 2 to this thesis.

In general, the explicit listing of lessons learned made the analysis of the authors' thinking easier. However, as in the interpretation of the interviews, I attempted to examine the conclusions which they drew in the context of some of the DM elements found in my earlier work.

One of Avison and Wood-Harper's lessons which provides support for the existence of implicit influences on CBSD practice in general terms is "Lesson 7 – the methodology is interpreted by users/analysts" (*ibid.*, p 267). Additionally, some of Avison and Wood-Harper's lessons could be linked back to deeper beliefs. For instance, "Lesson 6 – In certain situations the methodology gives insufficient

guidance" (*ibid.*, p 267) can be viewed as a result of the inductive Principle of Uniformity of Systems (see Chapter 6 below); given that the future is unpredictable, it is impossible for any predefined mechanism to give sufficient guidance for all circumstances. This lesson also supports the contention that Avison and Wood-Harper support the idea of the introduction of additional or replacement mechanisms into a CBSD process as required[47], suggesting that they believe that 'different notations used in modelling produce sufficiently similar results for substitution of one for another to be permissible'.

5.5.2.5 Conclusions

Direct support for the contention that the DM of CBSD exists is provided by the developers of Multiview when they conclude "Lesson 7 – the methodology is interpreted by users/analysts" (Avison and Wood-Harper, 1993, p 267) and "Conclusion 2 – Defining an information system is contingent" on, *inter alia*, "the information systems development team" (*ibid.*, p 269).

They also note that an unco-operative user "...contradicts the arguments of 'pure' Multiview ... in which *it is assumed* that it is always possible to use responsible participation in information systems development." (*ibid.*, p 267, my emphasis). To Avison and Wood-Harper, there is thus at least one assumption which underlies their method. I suggest that the set of such assumptions, both explicit and implicit, is in fact the DM which forms part of the Kuhn-based model presented in this thesis.

As in the previous investigations, evidence for the existence of a general belief system was also found in the form of a number of specific beliefs, either in this source alone or common to this and the other sources examined. These beliefs are summarised with the other results in Appendix 2. They include beliefs concerning the nature of action research as well as those related specifically to the development of computer-based systems.

It should be noted that other CBSD methods have been built or modified as a result of what is effectively action research, even if it has been in a non-academic environment (see for example OMT (Rumbaugh *et al.*, 1991)), and this is often seen as being a positive aspect of such mechanisms.

---

47 This actually happened with the modification of Multiview's function diagrams into structure charts in the book's case study 6 (*ibid.*, p 244).

5.5.3 A Report into a Failed Project

5.5.3.1 Introduction

It has also proved to be instructive to look for evidence to support the existence and content of the DM of CBSD by looking at a report of a development project which failed.

It is inevitable that many projects which fail will not be reported at all due to commercial or political embarrassment, leaving the analyst to examine nothing more than partial (in both senses!) reports in newspapers and books. However, sometimes reports into projects which failed are made public, and the example of the London Ambulance Service support system (LAS, 1993) has been examined.

5.5.3.2 Background: The Failure of the System

The following information, which is intended to provide a context for the lessons to be drawn from the London Ambulance Service ("LAS") report (LAS, 1993), is summarised from the report itself.

There is a national standard for the time taken for an ambulance to arrive after a call to the emergency services is made. It was felt by LAS management, and agreed by the inquiry team, that in order to meet this standard in London a computer aided dispatch system is needed, to co-ordinate calls for ambulances with the location and status of each vehicle.

An earlier attempt to provide such a computerised system having failed, LAS were using a manual paper-based system. At a time when industrial relations were very poor, LAS management decided to implement a new computerised system. They selected a small software house as prime contractor for the system on the basis of the lowest tender. Unproven technology was used to provide automatic notification of sometimes inaccurate location information on the ambulances. The system was implemented to a fixed, short deadline, without an attempt to gain the cooperation of the end user staff involved. As the system was developed, the software house accepted many requests for changes in requirements without proper authorisation. The systems software used to develop the system included unreliable versions of bought-in products, such as Windows 3.0, quickly replaced by most users by version 3.1, and the first production version of a new language,

Visual Basic 1.0. The inquiry team note that Visual Basic was employed for speed of development rather than speed of running, whereas the latter was actually needed in the computer-based system (*ibid.*, para. 3128). Staff were trained too far in advance and therefore forgot the new procedures before they used them under live conditions, and in any case the system had changed from the training system before it went live. Testing and change control of the system were inadequate. The system was unable to be run at an adequate speed, a problem which the inquiry team blamed on the software design rather than hardware deficiencies.

Finally, the system was made live in one stroke without the back-up equipment, necessary to support failures in the main computer, being installed let alone tested. After a week, the computer system crashed due to memory filling with unreleased memory allocations arising from a programming error (*ibid.*, para. 4039). LAS staff were forced to revert to the earlier manual system to maintain service, amid much embarrassment.

An inquiry was set up into the debacle. The inquiry team concluded that "Various points ... [are] raised about the quality and speed of the CAD[48] software. The quality issues are significant and are caused by time pressure to deliver, inadequate testing and poor quality control." (*ibid.*, para. 3120)

5.5.3.3 The Investigative Mechanism

The procedure adopted for this work was similar to that of the previous examinations of printed works. The inspectors' report (LAS, 1993) was read and trawled for explicit and implicit beliefs, as expressed in the text, which form part of the DM of the authors, and of the computer-based systems developers whose work was being reported on.

The results of this work are summarised in Appendix 2 of this thesis, in the form of DM elements for which evidence was found. The degree of overlap found between elements found in the LAS report and in other sources suggests that the same DM structure might apply to the LAS report as well as the others.

---

48 computer aided dispatch

5.5.3.4 From the Horse's Mouth ...

I have attended a presentation by one of the inquiry team members (Paul Williams[49]) on why computing projects fail, at a seminar organised by the London Society of Chartered Accountants.[50] This provided an opportunity to hear the views of a person who was knowledgeable about the particular project and its failure, but who saw the work from outside the discipline of CBSD.

His ultimate conclusion was that projects which fail do so due to bad management, not technical reasons. He believes that it should be possible to manage round any technical failures or problems which appear during a CBSD project. This provides a different perspective on the work of developing computer-based systems, and throws into sharper relief the beliefs listed in Appendix 2.

5.6 Conclusions: What Was Learned from the Investigations

The following conclusions which support the Kuhn-based model of CBSD can be drawn from the investigations.

5.6.1 Evidence for the Kuhn-based Model in General

- A set of DM elements seems to exist which unifies all of CBSD theory and practice. I have found evidence of parts of a generally-accepted set of beliefs common to many sources, in addition to those which divide the discipline into schools.

  Failure to find in any particular source an element which is believed to be common to all CBSD theory and practice, and is reflected as such in the list in Appendix 2, may be due either to a defect in the investigative mechanism, which was not intended to be this rigourous, or to an element not being made sufficiently explicit for me to find it in that source. This may apply particularly to those elements which are deemed to be 'obvious' to the author or subject concerned.

---

49 At the time Chairman of the IT Faculty of the Institute of Chartered Accountants in England and Wales, and Computer Audit partner at BDO Binder Hamlyn.

50 Avoiding Management Disasters in the Computer Environment; London Society of Chartered Accountants; 9 December 1993: my notes of an unpublished seminar, which was an informational rather than research presentation.

- Beliefs which people actually apply to their work are in some cases the same as are taught, even if the people applying them have had little formal training in the theory.

5.6.2 Evidence for CBSD Method Development being Analogous to Pre-science

- Differences in beliefs exist between method developers, which might form the basis of identifiable schools of thought.
- In some cases, different values can and do exist for the same belief. Consider for example the different explicit values for the results of the shared belief that a specific computational model should underlie a specific notation.

5.6.3 Evidence for CBSD Method Use being Analogous to Normal Science

- There is a set of beliefs underlying each example of CBSD theory and practice, which can be identified separately from that theory or practice.

  This set of beliefs can be identified, for example, by considering the influences on what the workers involved regard as the 'quality' of the process and the product of that work.
- Given that people try to develop high-quality systems, and with evidence pointing to developers using the belief systems implicit and explicit in their chosen or imposed development mechanisms as a basis for defining quality, the DM underlying a CBSD method has an effect on the systems developed.
- There is some explicit degree of trading off between elements to achieve the desired mix for a particular development project.

  The different definitions of 'quality' observed in the different sources exhibited trading off between lower-level criteria. This will be expanded on later in this thesis (see Chapter 7, below).
- There are disagreements about how to weight some elements between CBSD practitioners in general, and between participants in a project.

5.6.4 The Null Hypotheses Reconsidered

With regard to the null hypotheses described previously, I suggest that the following has been learned from the investigations:

- that *a identifiable DM does not exist in CBSD* has been put in doubt by my success in finding common aspects of belief across all areas of CBSD, from both theory (textbooks) and formalised and informal practice; and
- that *the individual elements of the DM identified during the book trawl cannot be identified in the practice of the discipline* has been disproven by my succeeding in doing exactly that.

5.7 Summary

In this chapter, I have described some initial investigations exploring the phenomena predicted by the Kuhn-based model of CBSD, and the conclusions which could be drawn from them.

The next step is to examine the results of the practical work, and see how they can be reflected in the Kuhn-based model. The next chapter will therefore extend and add detail to the theory presented previously.

6 Articulating And Exercising The Kuhn-based Model

6.1 Introduction

On the basis of the investigative work described in the previous chapter, we are now in a position to expand the Kuhn-based model of CBSD into more detail, especially with regard to the structure and content of the DM of CBSD.

Having done so, this chapter then:

- considers a small number of elements of the DM in detail, as examples of how the work resulting in the list of elements in Appendix 2 was performed, and how the Kuhn-based model can be made directly useful; and
- considers how the Kuhn-based theory of CBSD might be extended.

6.2 Structure of the Disciplinary Matrix

6.2.1 Introduction

This section describes and discusses the structure of the DM as determined by consideration of the results of the investigations.

Features introduced to the Kuhn-based model of CBSD include the division of all elements found to date into those which are, or appear at present to be, common to all computer-based systems developers, and those on which opinion in the global community of CBSD method developers is divided, forming the schools of thought referred to previously. This both sustains the contention that such schools exist, and supports it with explicit examples.

6.2.2 Dividing the Disciplinary Matrix – Common Elements and Schools

6.2.2.1 Observations from the Investigations

An examination of the elements identified in the DM of CBSD in the investigations shows that they can be divided into the following categories:

(1) those which are common to all computer-based systems developers;

(2) those which are common to all computer-based systems developers and developments, but which require that one, or a mixture, of a set of possible values be taken, such as the computational model used for modelling a design, which might perhaps be database, procedural, object-oriented or some other; and

(3) those which are optional for any computer-based systems developer or development activity, and which require that one, or a mixture, or a number of values be taken.

The second and third of these criteria, by reason of their possibly different values, divide computer-based systems developers and the activity of CBSD into *schools* according to the different sets of beliefs underlying their actions. This observation provides a rationale for this division of the discipline, which was noted earlier in this thesis.

The second group has been treated in Appendix 2 by including in those denoted as being common to all computer-based systems developers a general belief which unites them on the issue, such as 'All software design is with respect to a particular, identifiable computational model'. The detailed beliefs which divide the CBSD theorists on such issues are either left for future investigations, such as the lowest-level computational models from which formal and informal CBSD notations are derived, or included as dividing beliefs, as in the case of top-down vs bottom-up model design mechanisms.

I have organised the elements into Kuhn's four types of elements within each of the common and dividing groups of elements. This reflects the importance of the division of elements into common and dividing in understanding the current state of CBSD. This division has shown that examples of each type of element described by Kuhn can be found, but that there are few symbolic generalisations. This last point is discussed in more detail below.

6.2.2.2 Theoretical Considerations

It is important, if disturbing, to note a logical problem in defining any element of the DM as being 'common' to all computer-based systems developers, which is that in making the statement that they are followed by *all* computer-based systems developers we are making an inductive assumption. However, in at least one case,

an element is included in the DM of CBSD due to logical necessity rather than subjective belief. This reflects the inductive nature of CBSD itself, which is discussed in detail later in this chapter.

In the spirit of Popperian falsificationism, I therefore state in respect of each of those elements which I have stated in the table in Appendix 2 to be common to all computer-based systems developers that it is *believed* to be common to all computer-based systems developers, i.e. I have not yet seen an example of a computer-based systems developer who disagrees with that statement. This is an equivalent position to Popper's view of the most certain status possible of a theory as being 'not yet falsified'. This position reflects both the 'scientific' aspects of the DM underlying the work presented here, and the inevitability of some personal input by the investigator into the research at this early stage.

In a sense, the beliefs described here as common to all computer-based systems developers form a definition of who is a 'proper' computer-based systems developer, as against a 'hacker', who will work under a different DM producing different sorts of results. This definition of who is a proper computer-based systems developer (or should we call this person a 'software engineer'?) based on the acceptance of a particular DM is as Kuhn predicts, since, as we have seen, for him a scientific community is defined by the DM held in common. Moreover, for Kuhn the definition of who is a 'scientist' is also based on a (presumably smaller) set of beliefs, values and so on, common to and shared between all scientists (Kuhn, 1970, p 177).

This discussion might be seen as somewhat academic when considering computer-based systems developers in the real world. We might consider that it is easy to identify who is, or is not, a 'proper' computer-based systems developer, and sort out the 'proper developer' goats from the 'hacking' sheep. However, the issue will become crucial when considering those computer-based systems developers who are not often considered to be 'software engineers', i.e. those developers who, commonly working in small teams or alone, use informal development mechanisms rather than predefined, formalised methods.

6.2.2.3 Returning to the Investigations

Where might direct evidence be found in the investigations to support the division into common and dividing elements, and to support a suggested status for an element?

The most likely place to find this evidence would be in the results of the second phase of the book trawl, since this was devised specifically to look for a set of predetermined elements in the totality of a number of sources. The later investigations, *viz.* the interviews and the examinations of Multiview (Avison and Wood-Harper, 1993) and the London Ambulance Service report (LAS, 1993), were more general trawls, designed to extend, as much as to test, the existing results. Only in the second book trawl was a pre-existing list of elements being deliberately checked against sources, rather than the sources being looked at with as open a mind as possible for whatever might be found. The results of this exercise are that, of eleven elements supported in the first trawl and described in Appendix 2 as being common to all CBSD activities, six elements were supported in all of the second trawl sources, and another four in three of the four sources. Given that the sources were being scanned for DM elements which may have been either overlooked by the authors, or regarded by them as 'obvious' and therefore not worth stating explicitly (this may apply in particular to the common element 'the system should be designed to run as fast as possible given any other relevant constraints', for which positive evidence was found in only one second trawl source), and that those cases for which a clear opinion was not found in a text are excluded from these figures,[51] I feel that the correlation found between an element being common and its being found in all or most of the second trawl sources is reasonable.

It should be noted in this context that much of the work on allocating individual DM elements to be either *common* to all CBSD workers or *dividing* them into schools has been based on my own knowledge, experience and introspection, justified where possible by the investigative work. Where, for instance, I believe an element to be common to all computer-based systems developers but it has not been found in all of the sources used for the second book trawl, such as 'elegance in systems design', I have maintained its common status in the table. On the other

---

51 Counting these unclear results from second trawl sources as supportive of the elements concerned would have changed one 'three' to a 'four', and the single case of support from only one source to that from two.

hand, three elements for which strong support was found in all of the second trawl sources, and for which no negative opinion was found in any of the investigations, *viz.* 'A quality-aware CBSD process will result in a higher quality product', 'The software development process *is* capable of being managed' and 'Controlling the software process produces benefits which outweigh the costs', have been included in Appendix 2 amongst those dividing CBSD into schools, since I regard these elements as not being essential for CBSD activities undertaken by an individual and/or in an informal manner.[52] The objective clarification of such subjective issues is an area for future work extending that presented in this thesis.

The significance of the second book trawl figures is, therefore, that any element found and agreed with in all of the sources during that work *might* be common to all computer-based systems developers. Alternatively, such a finding might reflect a commonality of belief amongst the authors concerned which is not reflected throughout the CBSD community, i.e. all of the authors belong to the same school on this issue.

Dividing elements may, by definition, be agreed or disagreed with by individual CBSD theorists or practitioners within the CBSD community. An element disagreed with in one of the second tranche books is likely to be related to an issue of importance, since it is reflected in at least two sources, but the disagreement indicates that differing schools of opinion exist concerning it. Such an element is therefore inevitably a dividing element rather than a common one, although the CBSD community might agree that the issue raised needs to be addressed. In general, evidence for the existence of such a belief in one school of CBSD theory or practice may be found in a particular source in the form either of agreement or disagreement with that belief, and this is reflected in the negative opinions concerning some dividing elements documented in Appendix 2.

6.2.3 The Fine Structure of the Disciplinary Matrix

Within the general division of the DM into common and dividing elements, I have further subdivided the elements identified to date under a series of headings which I consider to be appropriate. Whilst some headings dividing the DM element list

---

52 A fourth element for which support was found in all of the sources used in the second trawl was denied by a source in the first trawl. I therefore regard this element, 'Computer-based tool support for software development is essential', as a dividing element.

in Appendix 2 only group together elements which are related by subject matter, others form hierarchical structures within the DM.

A consideration of this discussion in the context of the higher-level paradigms identified by others[53] is suggested as a future extension to this work.

As an example of a hierarchical set of elements, consider those DM elements related to the decomposition of models of problems and solutions into sub-models. The connections documented in this hierarchy form a series of conditions in which the higher-level elements are *necessary* but not necessarily *sufficient* to support the lower-level elements. I suggest that it would be unlikely that a person would believe in, or at least *act* as if he or she believed in,

> 'top-down decomposition is better than bottom-up composition'

in respect of a software design without believing in, or acting according to a belief that

> 'computer systems should be built up from components rather than being monolithic single programs'

The former belief does not make sense if the latter is denied. There is therefore a practical necessity in a computer-based systems developer believing in the latter before the former is accepted.[54,55]

I believe that the hierarchical connections between elements exemplified here is one aspect of the inter-relationships between elements. In addition to these direct connections, it should be borne in mind that no work has yet been performed to establish whether there are less obvious connections between elements, possibly *across* the hierarchies described here. In order to draw any conclusions as to the existence or otherwise of such connections between pairs of elements, it will be

---

53 See Chapter 8, below.

54 Reasoning of this nature has formed the intellectual basis of much of the introspection performed during this research.

55 I have not considered in this research whether the relationship is one of *logical* necessity; reaching this conclusion would require a logical analysis of each of the sets of statements. This may form a useful direction for future research, once the content of the DM has been established with some degree of assurance.

necessary to consider each pair separately, and perhaps to perform experiments using the two as axes.

As an example of a group which lacks an obvious finer hierarchical structure, consider the set of 'beliefs about group and individual development, and staffing computer-based systems development programmes' which divide computer-based systems developers into schools. These beliefs seem only to be connected by being in the same category, which effectively forms a bucket for these beliefs. Such a categorisation is merely a useful way of subdividing what would otherwise be a long undifferentiated list, but for these cases other subdivisions can be devised which will serve the purpose as well as those which I have presented.

6.3 Extending the Theory

6.3.1 Introduction

In this section I consider some aspects of the results of the investigations, and extend the Kuhn-based model in the context of these results.

6.3.2 Why are There So Few Symbolic Generalisations?

Kuhn, using physics as his exemplary science, expects there to be a number of symbolic generalisations in a science's paradigm, some of which are highly important in describing the underlying theories. An example which he gives is the succinct but complete description of the Law of Motion as '$F = m\,a$' (Kuhn, 1970, p 183).

On examining the detailed results of my investigations, a question which arose almost immediately was that of the small number of symbolic generalisations in my list of elements, when compared with other types of elements. Those which do appear are either general necessities of computation such as the Church–Turing thesis and rules of complexity and computability, or are dividing elements related only to the *management* of the process of CBSD, such as the COCOMO model (Boehm, 1981), rather than to the process itself. Why have so few symbolic generalisations been found in this work which are unique to the activity, as against the management or the underlying theory, of developing computer-based systems?

Possible reasons for this might be any or all of:

- my not looking in detail at those schools of CBSD which emphasise mathematically-based work, such as formal methods – this is considered under 'future work' below, although since this is only a school of the discipline the results would not be common to all computer-based systems developers;
- the lack of such symbolic generalisations in the knowledge base of today's computer-based systems developers, perhaps because the science base has not yet been developed to the stage at which such quantitative theories have emerged; or
- the natural difference between a science discipline and a design discipline, which might conceivably result in different-looking DMs.

Consideration of this issue is included under 'future work' below.

6.3.3 Musings on 'Quality'

One of the strands which have united the textbook trawling and the interviews described above is an examination of how CBSD theorists and practitioners define and think about 'quality' in a computer-based system. Does it seem reasonable to include an element in the DM of CBSD which reflects this position, such as 'computer-based systems should be of high quality', without defining what 'quality' is?

Since this thesis concerns the proposal of a philosophical model of CBSD rather than constituting an investigation into software 'quality', I have not examined published definitions of 'quality', nor looked beyond the work the results of which have been presented above. However, can anything be learned from the investigations described in this thesis? In addition to providing support for the existence of the DM by locating specific elements of it, can we draw any additional conclusions about 'quality' in a computer-based system from the investigation, such as how it is related to its underlying beliefs?

One immediate conclusion is that we cannot currently identify any single, complete, unambiguous definition of 'quality' in a CBSD process or product with which all of the authors and interviewees would be in absolute agreement. This is in itself important, because it might weaken the claim that such a belief as 'computer-based systems should be of a high quality' should form a part of the

DM. If the term 'quality' is incapable of being defined to the agreement of all, then of what use is it to say that computer-based systems developers believe, or should believe, that 'computer-based systems should be of high quality'? The agreement is purely syntactic, without any reference to the meanings of the symbols agreed to. They all agree, so long as they do not inquire too deeply into what it is that they agree upon.

It can be argued that making this belief explicit in a DM would be of use despite problems in defining the key term. This is because its opposite, i.e. that 'computer-based systems do not always have to be of high quality' is a rational position to take when confronted with, say, the need for a utility to be used once only and then discarded. However, I disagree with this view, since I suggest that the example given only shows that 'quality' is relative to, *inter alia*, the circumstances and requirements of the system under consideration. In the case of a single-use utility program suite, it still has to do its job correctly and without using so much resource that its work cannot be completed. A measure of 'quality' can therefore be constructed for this occasion, and I believe that this can be done for any other occasion. Whilst all computer-based systems developers do (or should?) agree that computer-based systems should be of high quality, the *definition* of that quality will depend, *inter alia*, on the circumstances.

The DM elements which relate to the quality of a CBSD process or product must therefore include:

> The definition of 'quality' of a CBSD process or product is related to the circumstances under which the work is performed.

In addition, the ability to reason about the quality of a CBSD process or product is based on the belief that:

> A CBSD process or product has a level of 'quality' which can be compared with that of other processes or products.

Neither of these definitions assists us in determining what 'quality' in a CBSD process or product is, but some reasoning about it in general terms, as well as the ability to rank CBSD processes or products in order of 'quality', is possible.

In addition to 'quality', there are other terms whose definition differs between CBSD schools. I have, for instance, included 'object-oriented' methods from two sources (Sommerville, 1992; Booch, 1994) as subscribing to an 'object-oriented' computational model, without enquiring too closely into the similarities and differences between the usages of the term made by these two authors. However I can conceive of a analysis based on decomposing the meaning of the term 'object-oriented'. I suggest that such detailed analysis of each of the terms used in defining each of the DM elements presented in this thesis lies outside the scope of the work presented here, and must await further time and resources. For object-oriented enthusiasts, such a decomposition of their favourite term into underlying beliefs and models might be of great use in helping to define the term and refine what object-orientation really means, but for those taking a wider view this may be a matter of detail.

Given that there are a number of terms in CBSD which are defined differently by schools of CBSD, the criterion of differences in defining terms might provide a heuristic for determining the level at which decomposition of a belief or model into its underlying beliefs or sub-models can be concluded for the purposes of consideration of CBSD as a unified discipline. At some point in the decomposition, a material divergence of opinion on some aspects of an element might appear, and at this point differing schools based on the divergences can be identified and investigated.

This consideration leads to the suggestion that it might be possible to examine the sub-disciplines or schools of CBSD by analysing their own DMs, forcing the adherents of, say, different schools of object-orientedness to consider the significance and usefulness of the differences between their schools' beliefs. Such an analysis is outside the scope of this thesis, but is included as a pointer to future work.

The conclusion that the generalisable aspects of the term 'quality' cannot be defined in a manner to satisfy all of the authors and interviewees considered above supports the Kuhnian pre-science model suggested for the current state of CBSD. A key term, *viz* 'quality', cannot be defined in such a manner as to satisfy all of them, in a manner analogous to the problem of defining 'planets' and 'stars' before the work of Copernicus. The question remains of whether a definition can be agreed of what constitutes the situation-independent aspects of software 'quality',

which is as clear, unambiguous and universally accepted as the current definition of 'planet' is to astronomers.

6.3.4 How Elements of the Disciplinary Matrix Interact or Support Each Other – The Strange Case of the Waterfall Model

There is no reason in the Kuhn-based model of CBSD for the DM underlying CBSD theory and that underlying its practice to be identical in all respects, or for the effect of an element of the DM of CBSD on its practice to be exactly what is expected by CBSD theorists. As an example of how the Kuhn-based model might be developed by building testable theories using elements of that model, as Popper would expect of a scientific theory, I present here an argument based on that model, and reflecting both the DM and the different roles involved in CBSD. The result, which I feel is at least a little surprising, is not one which is to be applied immediately, but a tentative conclusion which is capable of being tested, and which I believe should be so tested.

Let us consider the possible effects of the desire to control costs and timescales in a CBSD project. This results in a method whose process model is constrained to eliminate feedback, reopening and reworking previous phases of work; i.e. the pure 'waterfall model' of the CBSD process (Royce, 1970).

We can observe that the waterfall process model is both used and criticised in current CBSD theory and practice. Van Vliet introduces it (1993, p 32 *et seq.*), in a version which includes feedback between its phases. One theme in recent CBSD theory is the stating of the obsolescence or evil effects of the feedback-free waterfall model of CBSD (see, for example, Gilb, 1985). Explicitly criticised and rejected in these criticisms is the lack of ability to feed back and repeat or rework earlier phases in the development cycle. Ignoring the possible implicit rejection of the view that all CBSD projects pass through a set of essential activities, whatever any particular method calls them, from requirements analysis – finding out what to do, through systems analysis – finding out how to do it, design and implementation – doing it, to maintenance and eventual replacement, it is pertinent to ask on what grounds the rejection of the waterfall model is based.[56]

---

56 It should be noted that the feedback-free version of the waterfall model itself also makes some assumptions, such as that goals and objectives of a development can be clearly and unambiguously defined in advance.

Looking at the set of DM elements identified above, we can see at least three candidates for the underlying cause for this rejection. These are:

- the result of *experience:*
  - experiments under controlled conditions;
  - investigative research; or
  - unformalised real world experience, i.e. discipline craft knowledge;
- the belief that the waterfall model is incompatible with some techniques found to be useful in CBSD, such as prototyping, or process models which rely on explicit iteration of some or all phases (such as phased development or the Collective Learning model (Winder *et al.*, 1992); and
- the dictates of *fashion.*

I suggest that any or all of these beliefs may affect the decision in any particular case to reject the waterfall model. However, making these underlying beliefs explicit in turn makes it easier to reason about the way in which the waterfall model fails to meet the needs of computer-based systems developers, and to consider whether the rejection of it is sensible in any individual case. In any case, the current DM of theoretical CBSD makes it difficult for a computer-based systems developer to argue in academic circles that advantages exist in the feedback-free waterfall model, such as the ability to determine the costs of completed phases, let alone that these advantages might outweigh the disadvantages for some projects.

As a side-effect of the waterfall model, it seems reasonable to postulate that a development mechanism which does not allow developers to rework sections of a design which is found to be inappropriate or unimplementable during the implementation phase might cause the developers to be conservative in their design decisions. Knowing that they cannot redo a speculative or innovative, and therefore risky, suggestion for implementation should problems emerge, they might shy away from such a solution and use what they are more certain to be able to complete. The only way to support an innovative solution would be to perform some test implementation, which might not be feasible within the cost constraints and timescales and which in any case goes against the spirit of a feedback-free waterfall process. The developers might have to 'hide' some trial implementation work to test their theories within the budget allocated to an earlier phase, since

they are not meant to be doing implementation work yet – the process model says so. Such work would be in direct conflict with the DM of the CBSD method in use.

A result of the decision to control timescales and costs might therefore have been to restrict the designers' freedom to innovate, or indeed to search repeatedly for a solution to a difficult problem. Was this what was originally required, or desired?

The process which has caused this suggested side-effect might be:

input: desire for control of costs/time of process; 'need' for hard management control (see Friedman with Cornford, 1989);

phenomenon: method with incomplete/limited feedback allowed;

output: conservative design decisions in software built using that method.

The effects of feasibility studies in reducing the effect described above should be mentioned here. However, I doubt whether they eliminate conservatism in designers completely, at least to the extent that testing for the effect would still be worthwhile.

The general conclusion which can be drawn from this argument is that it cannot be assumed that a hoped-for effect from a method design decision will be the actual effect, and that such decisions will be free from unexpected, and perhaps negative, side-effects. We need to have a mechanism for investigating and checking how a desire on the part of a method designer to engender a particular attitude in a computer-based systems developer is implementable as expected and without side-effects, before we can incorporate a new concept in a method with confidence.

Also, the conclusion seems to support the contention that method testing should involve people outside the method development team to ensure that as many as possible of the method's assumptions are explicit, not implicit, in the documentation of that method. The Kuhn-based model of CBSD implies that any such testing cannot be carried out using people lacking direct experience of a method, such as students. The latter are under pressure to use a method 'as is', are new to it so that they will be more likely to check their knowledge with the

manuals,[57] and lack the knowledge and experience which can only be gained from involvement in developing other systems. Additionally, this conclusion suggests that the outcomes of action research programmes need to be blind tested by outsiders before such results are accepted unreservedly.

6.3.5 Evidence for Personal Paradigm Shifts

How much of the theory described concerning the possibility of changes in personal belief systems, referred to in Chapter 4 above as personal paradigm shifts, can be supported either directly by the investigations reported in the preceding chapter or by ideas arising from a close examination of the sources used for those investigations?

Perhaps the best evidence arising from my investigative work for the influence of personal paradigm shifts on CBSD is in the lag between the explicit acceptance of a new computational model and the making of changes to other aspects of thinking which might need alteration to match it. I instance Schach's book (1993), in which an exposition of cohesion and coupling, measures developed to analyse *procedural* designs based on Myers (1978), which predates 'modern' object-oriented methods, is followed by consideration of more recent ideas of data encapsulation (*ibid.*, pp 254–262), abstract data types (*ibid.*, pp 262–265), information hiding (*ibid.*, pp 265–268) and objects (*ibid.*, pp 268–271; his advocacy of objects as agents of reuse is striking) which do not mention either cohesion or coupling.

The lack of coherence between the 'old' thinking, still firmly rooted in a procedural model, with the 'new' implicit in the more recent computational models, is noticeable. He makes references to cohesion and coupling in the context of data flow analysis, but these measures of quality have been lost by the time objects are considered. I contrast Sommerville, who has modified the cohesion measure to allow for 'object cohesion'. To me, this suggests that he has to some extent merged the underlying computational models in his mind and therefore performed at least a partial personal paradigm shift, whereas Schach has described first one then the other without the unconscious conversion.

---

57 Based on my observation and experience of CBSD practice, the DM in Appendix 2 includes the element 'When all else fails, read the manual'.

The evidence for Schach not having shifted his unconscious paradigm from procedural to object-oriented, despite his espousing of objects as a good thing, might perhaps be immediately due to:

- my working on a textbook which has been updated without a complete rewrite, adding newer computational models without altering aspects brought forward from the earlier edition which relate to the older model. This may be due either to a combination of time and market pressures to produce a new edition of the book with the relevant terminology but without time to modify the book completely, or to a lack on Schach's part of the perception of a need to perform that updating. The latter at least supports either the suggestion of the lack of a shift in his mind, or his having performed such a shift and his rereading the section on cohesion and coupling in that new light without perceiving that changes are needed; and/or
- Schach thinking in terms of one computational model, but writing in terms of another in specific parts of the book to meet what he perceives as (or is told are) the market's requirements, i.e. Schach writing the book for a course or courses which require specific aspects of CBSD to be taught in a specific manner (*teach* cohesion and coupling, then *teach* objects), in which case the confusion in computational models is due to those who have demanded the content of his book rather than Schach himself, although he has not freed himself from that confusion.

To my mind the latter explanation is supported by the lack of detail on specific object-oriented methods in his description of the design phase, and the lack of any discussion of objects as an analysis modelling tool (but note that Structured Systems Analysis is included). In this light, Schach's identification of object-based design with software reuse, another current 'hot topic', suggests that fashion might be being followed in his introduction of object-oriented design, and that either he or his market, or both, is following it.[58]

The impression left on me is one of the book trying to face in two directions simultaneously, without having reconciled these two positions. I suggest that for Schach the 'need' for a change in mind-set may have been perceived, but the

---

58 The influence of fashion on CBSD theory and practice is discussed in more detail below.

change itself seems not to have been internalised. He has not yet experienced a personal paradigm shift.

6.4 Some Individual Elements of the Disciplinary Matrix Considered in Detail

6.4.1 Introduction

This section considers in depth one element of the DM of CBSD, the Principle of Uniformity of Systems, which has arisen as a result of questioning what underlies the development of computer-based systems using predefined mechanisms. In addition, two other elements, the effects of fashion, and the external aesthetics of computer-based systems, are discussed in outline.

The identification of these elements has arisen directly from the consideration of the theory and practice of CBSD in the context of Kuhn's work. The Principle of Uniformity of Systems is an example of an implicit, objective element influencing CBSD, which I have identified by thinking about CBSD from first principles using the idea that a belief structure in the form of a DM exists as a starting point. The other elements considered in less detail have been identified by examining parallels with other disciplines or performing investigations with the image of a DM in mind.

6.4.2 The Principle of Uniformity of Systems

6.4.2.1 Introduction

In this section I place the work of the computer-based systems developer in the context of an important element of the metaphysical underpinnings of all scientific theories, the Principle of Uniformity of Nature. This principle provides a basis for using predefined methods and tools, but at the same time introduces unavoidable weaknesses into the practice of CBSD. The work presented here is based on a paper presented at a conference during my PhD studies (Winder and Wernick, 1993).

The inductive aspect of the principle of uniformity of nature controls the assumptions underlying CBSD practice, in a way similar to those of a scientific discipline. The relevance for the practising computer-based systems developer lies in the fact that one cannot be sure at the start of a development project whether

the tools and techniques initially selected for the work involved are appropriate for the task, and therefore one must always be aware that a need may arise for these tools or techniques to be changed or modified as the project proceeds. Moreover, methods and tools used by a computer-based systems developer may need to be sufficiently flexible to accommodate this change.

The inductive principle underlying the predefined tools used for CBSD is not unique to this discipline; it is common to all design disciplines. However, the degree to which the principle affects the practice of a design discipline is dependent on the degree to which that practice is based on predefined mechanisms, and CBSD is very well supplied with such mechanisms in its tools, techniques and methods.

6.4.2.2 The Place of Inductivism In Science: Inductivism Introduced

Inductivism is a logical mechanism, which may be defined informally as 'learning from experience', whereby if an event has occurred in the past following on from certain other events, it is assumed that it will always occur in the future after these same events.

The scientist may propose a general law, based on a number of experiments, such as 'Metals expand when heated' (see Chalmers, 1992, p 4, for the source of this example). The researcher will have performed, or read reports of, many experiments which support this contention, and his or her confidence in the correctness of the proposed law will be increased if it can be explained by a theoretical mechanism, rather than merely supported by an appeal to the weight of evidence or number of positive experimental confirmations. It can be shown that this position, although it forms the basis of much 'common sense' thinking, and is claimed to be the basis of some scientific discoveries, is indefensible in logical terms; see, for example, Chalmers (1992, pp 13–17).

However logically indefensible the inductivist position may be, without some assumption that the same rules apply everywhere, we are unable to make general statements about any phenomena we may encounter other than those which we have observed directly in the past. In devising general laws, the scientist is faced with a major problem. While these laws may lay claim to universal validity, they can only be based on a limited number of observations.

This problem is addressed pragmatically by stating a general principle whereby the results of finite numbers of observations can be generalised to produce universal laws. This idea is so fundamental to philosophical and scientific thinking that it has been formalised as the Principle of Uniformity of Nature.

This Principle may be formulated as follows by the philosopher:

> "As certain uniformities – those we believe to constitute genuine laws of nature – have occurred in the past, so they will continue in the future." (Hospers, 1970, p 254)

or by the scientist (or, more accurately, the Natural Philosopher) as:

> "The qualities of bodies ... which are found to belong to all bodies within the reach of our experiments, are to be esteemed the universal qualities of all bodies whatsoever." (Newton, 1726, p 398)[59]

6.4.2.3 The Inductive Nature of Computer-based Systems Development

What has the inductive nature of science, and in particular the Principle of Uniformity of Nature introduced earlier, to do with CBSD?

At the heart of any predefined method or technique proposed for developing computer-based systems, there are two major assumptions:

1. that the universe in which computer-based systems are developed, which is made up of the computing environment (comprising software and hardware for development, maintenance and live running), the underlying belief systems of the system's developers, and the user environment, will be the same for the next system which we will develop as it was when the method was designed and when we developed our last successful similar system using that method or technique; and
2. that the properties of the *application* to be developed, and of the *environment* in which it will be developed – namely the languages used for analysis, design and implementation and the organisational context within

---

59 Note that Newton's fourth rule of reasoning (Newton, 1726, p 400) allows changes in a theory to be made if the inductive assumption is subsequently found to be incorrect in the light of new observations. However, Kuhn's work suggests that this flexibility is lost in the case of the fundamentals of a theory due to pressures to maintain the DM as inviolate.

which the development is taking place, are sufficiently similar to allow the development a good chance of success.

As with the domain of application, the choice of a method for developing a system also relies ultimately on the two inductive hypotheses given above, which attempt to correlate the universe of the method with that of the system to be built.

For the informal developer, applying only the knowledge based on his or her experience and reflected in his or her DM, rather than employing that embodied in a predefined method, that application is as inductive at heart as that of the method user. However, the lack of formality may allow greater allowances to be made for a failure in an underlying inductive assertion.

To generalise these assumptions, as an analogue to the Principle of Uniformity of Nature, I propose the Principle of Uniformity of Systems, which, expressed informally, states that:

> *The next system we build will be enough like the last one we built of this type, and those we or others built to test the tools we intend to use, to allow us to use the same DM and tools to build it.*

If any of the elements forming the universe in which the development of a new system is taking place differs from those assumed in choosing the method, or which have resulted in the DM which will be applied, the developers may need either to change the method to match the changed circumstances, or to modify some elements of the universe to match the method. However, *it is impossible to determine whether such a change is required until the mismatch is observed, probably when a problem arises somewhere during the subsequent progress of the development process.* The problem may not be detected immediately it occurs.

Even if a computer-based systems developer can, by reason of experience, select a method, build an *ad hoc* method from previously-known techniques and models, or apply previous experience to devise some way to develop a computer system, there is still a set of assumptions in his or her head which guides this selection. It is likely that the more familiar the computer-based systems developer is with the problem domain, the greater is the probability that the match between the

universes will be acceptable. Nevertheless the Principle of Uniformity of Systems will still stand in the background, guiding the process of method selection and use.

The assumption concerning the relationship between the universes of method and system, and the Principle of Uniformity of Systems which formalises it, are purely inductive, like the Principle of Uniformity of Nature. However, as we have seen, the adoption of inductivism as a basis for reasoning is not logically defensible. The only justifications for its adoption in this case are that it has worked so far – itself an inductive argument – and that without it we cannot proceed to formalise CBSD at all, in the same way as the scientist cannot start to think of universal Laws of Nature without the Principle of Uniformity of Nature. We can draw a direct analogy here; the mechanisms of working of CBSD *require* the Principle of Uniformity of Systems in the same way that scientific theory as we know it could not exist without the Principle of Uniformity of Nature.

Dasgupta (1991, p 158–160) points out that the fundamental analyse-synthesis-evaluate model or *design paradigm*[60] of how to develop computer-based systems is based on inductivism. One must first accumulate information about the world, then infer a theory to design a system. Fortunately, the risk associated with the inductive step is less for a design discipline than for a science, in that the designer assumes that the inductively-derived rule will apply to the next single entity to be designed and built, whereas the scientist is attempting to devise natural laws with universal applicability.

6.4.2.4 Where Does Inductivism Apply in Computer-based Systems Development?

Inductivism operates in several areas in CBSD:

- the development of methods and other tools;
- the theory and model building which occurs as part of the CBSD process, when predefined methods or personal experience are applied;
- the writing of programs in languages chosen before the development of the systems starts, in which the computational model of the language is assumed to be suitable for expressing the required real-world model; and

60 Dasgupta's design paradigms will be discussed later in this thesis (Chapter 8), as part of a review of related work.

- the development and design of packaged software and of virtual machine environments for specific applications.

I now consider each of these in turn.

Building Tools for Developing Computer-based Systems

The Principle of Uniformity of Systems is applicable to the development of CBSD methods, in so far as all method builders assume that the systems which they build with their method will be sufficiently like the example system models which they have used in developing and testing their method to allow the same method to be used in both cases. As noted above, this is pure inductivism.

A consequence of this observation is that any method may need to protect itself against a failure of inductive reasoning, to allow for circumstances in which the inductive assumptions implicit in the method and the CBSD process are shown by later work to be inapplicable in this case. This protection can be achieved by allowing some measure of feedback from one stage of the process to any of the preceding stages.

For this reason, I suggest that any method, or process model underlying a method, which does not allow feedback to occur freely places its users in danger of producing inappropriate systems, or indeed of failing to produce systems at all, for purely logical reasons *independent of any other qualities of the method.* As a generality, those for whom the system is being developed should perhaps be warned that such problems may arise, as a counterpoint to the more immediately apparent managerial advantages of such approaches, which I have instanced previously when considering the waterfall model.

Developing Computer-based Systems

Inductivism also underlies practical CBSD work. The Principle of Uniformity of Systems is applied here to support the use of methods, techniques and DM found adequate for previous systems to a new, and therefore usually to some extent unknown, environment. Computer-based systems developers must be aware of the danger of this inductive assumption, and be ready to correct any problems which may arise. These problems may appear at any stage in the process, when a model produced as part of an earlier stage becomes at worst unusable, or at least obviously less than optimal, at a later stage. For example, we cannot be *certain*

that we have used an acceptable analysis modelling language, whether graphical or verbal, in a particular situation until the results of the analysis have been successfully implemented and installed, and the system has survived the subsequent period of maintenance. The same argument applies to design and implementation languages.

As before, this requirement can be addressed by the adoption of a formal or informal process model which allows unrestricted feedback from any stage of the software life cycle to any previous stage. Additionally, a method which offers a range of possible modelling techniques, or which acts as a framework within which such elements can be mixed and matched as required to meet the individual circumstances, will offer more scope for the computer-based systems developer to tailor his or her work to the situation. Such a method therefore makes fewer inductive assumptions, and provides mechanisms for the reworking of stages of the process after a failure of an inductive step. The choices made by the computer-based systems developer from the options available will, however, still be inductive at heart, relying on his or her theorising about the future from past experience.

Selection of Programming Language

The Principle of Uniformity of Systems leads to the conclusion that any choice of implementation language is based on induction from existing knowledge applied to the actual problem domain. Therefore this choice should be subject to constant review and re-evaluation during the implementation cycle, and perhaps later, as exemplified by the demand for re-implementation work from consultancies.

Additionally, since the danger of inductive failure becomes greater as the size of the inductive step increases, the earlier this choice is made in the development cycle, the more it should be subjected to scrutiny.

Developing and Using Pre-written Software

Any packaged software product, such as an application package or a reusable software library of components or routines, reflects aspects of its developers' DM. These might, for instance, include views as to the suitability of particular computational models for the problem being addressed. The adoption of such a software product also reflects in its users the inductive assumption that the models underlying the software, and their equivalents in the problem being addressed, are sufficiently similar to allow the use of that product to succeed.

Some pre-written software products, in the form of language implementations, application packages or libraries, attempt to cover all of a particular application area, or all application areas. In this attempt, those developers who claim broad applicability are applying inductivism in their design as they seek to extrapolate from their existing experience to future situations. They cannot know the situations in which their product will be used in the future, but they assert that their product will be usable in them. This comment applies equally both to general-purpose computer languages and commercial systems development support utilities, and to software systems for specific applications.

The risk associated with the inductive step here is that 'generally-applicable' software will not work well in a particular environment or application. As the knowledge of the environment increases, so the certainty of application increases and therefore the risk decreases. However, that risk is still present.

The inductive nature of the underlying assumptions of any computer-based system trying to be in any way 'general' is clear. The practical result of the possibility of a logical failure of the inductive process must be that any application of generally applicable computer-based systems should be made on the basis of trials and of careful consideration of the alternatives. This may include the development of bespoke computer-based systems if the compromises required to meet the assumptions underlying the pre-written software are considered to be too great.

6.4.2.5 What Effect Do The Inductive Assumptions Have on Computer-based Systems Development Practice?

As yet, I am unable to make any statements concerning the *actual* effect which the inductive nature of computer systems development has on its practice. No research has been conducted to determine how often inductive failure occurs, or as to the nature of an 'inductive failure'. Is it black and white, pass/fail? or more likely a shading off from the optimal, down to – eventually – complete project failure? It is also not yet possible to state what the actual cost of inductive failures may be in practice, and therefore the degree to which the costs of providing safeguards against such failures can be traded off against benefits.

However, the Kuhn-based model has pinpointed a theoretical basis for research to start into the effects of inductivism on CBSD practice.

6.4.2.6 Conclusion

The activity of method-based CBSD as currently practised is fundamentally grounded in assumptions which can only be supported by using inductive reasoning. This inductive logic needs to be reflected in the DM of any CBSD activity in which previous experience or existing tools are used, which effectively encompasses all such activities.

By making explicit the ever-present implicit assertion that the Principle of Uniformity of Systems is in effect during CBSD activities, I suggest that I have demonstrated that research based on the Kuhn-based model can produce useful results which CBSD theorists can apply in their method and tool building, and which CBSD practitioners can apply in their development work.

6.4.3 The Effects of Fashion

I have suggested in the list of DM elements in Appendix 2 that the dictates of fashion form an influence on CBSD methods and their authors. This implies that, as suggested earlier in this thesis, not all changes by all method developers are dictated by certainty of improved results. Rather, it is the case that some changes are made because an appearance of modernity is necessary to retain credibility in the method and its authors. To retain faith, a method must not be seen to espouse 'out of date' mechanisms and/or models. The influence might be due to peer and/or market pressures.

As an example of this, 'object-oriented everything' is, at the time of writing, the current fashion. As a direct result, people claim this attribute for methods, tools or techniques under the most dubious circumstances, using it as a marketing ploy and diluting or corrupting the definition of 'object-oriented' along the way. As I have suggested above, an object-oriented computational model seems to have been bolted on to one of the sources used for my investigative work (Schach, 1993), either without modifying the underlying thinking, or without the perception that other aspects need to be changed to match the new thinking.

Another recent example of fashion in CBSD is that of the trend towards a CBSD management approach which emphasises Quality Assurance and quantitative metrics for process and product 'quality'. Jackson (1983), writing a decade before

van Vliet (1993), does not have a chapter or even a section in his book on 'quality' or 'quality assurance'. I am certain that he was aware of the need for computer-based systems developed using JSD to exhibit some aspects of what he would call 'high quality', but there is no explicit appeal to the need for it. Advice is sprinkled through the book on how to use JSD well, and exemplars are provided in three appendices, but Jackson's index does not even have an entry for 'quality'.

Contrast this with, say, van Vliet's (1993) approach ten years later, in which a part of the text is devoted to Quality Assurance and praise for quantitative measurements of 'software quality'. 'Quality' in software systems has become explicit, and can and must be measured. However, as we have seen, any definition of 'quality' in software can contain subjective aspects, and any attempt to reduce it to quantitative measures may therefore be problematical.

Despite this, a number of such measures have gained acceptance in sections of the CBSD community; see, for example, the account of cyclometric complexity in Fenton (1991, pp 279–281). However, current work on quantitative measures has resulted in some examples which seem to strain to find quantitative measures, rather than using qualitative, perhaps comparative, mechanisms. As an example of the potential effects on mind-set of a desire to quantify 'quality', I cite a measure used to demonstrate a "quality attribute definition that can be used by both users and developers" (van Vliet, 1993, p 80, Fig 6.3). This is a definition of "friendliness" in a computer-based system as being that combination of features which allows a user new to that system to learn, in less than one day, to complete 90% of tasks successfully in no more than twice the time of an experienced user. Whilst this provides a clear measure, and a definite system success/fail criterion based on the results of tests with a number of subjects, I am concerned that this quantitative measure has been called "friendliness". Has the desire for a quantitative measure on the part of this attribute's specifier led to him or her forgetting what the term "friendliness" actually means?

A compromise view of 'quality' is seen in Booch's work (1994), in which a Quality Assurance and control-based approach is modified by a perception that design is an artistic activity which cannot be controlled in the manner suggested. Booch explicitly states that a compromise is needed between the informality needed for good software design and the formality of the suggested management approach. There is an input here from the current quantitatively-oriented management

fashion, but resistance from, perhaps, a DM which has not yet undergone a personal paradigm shift.

An interesting example of fashion (and the continued focus of method developers on larger-scale projects) is the nature of computer tool support assumed by the authors in the second phase of the book trawl. As has been seen above, all of the four authors assume that such tools will be available, but of them Jackson, writing ten years before the others, assumes that nothing more than some sort of drafting tool will be available to help keep diagrams up to date easily. The others require more functionality from their support software. Is this an example of invention becoming the mother of necessity, in that once a CASE tool has been devised then all method developers must assume that it will be present to support their work? It seems that the idea of manual record-keeping systems, which may be adequate for smaller systems, are not felt to be enough for the modern computer-based systems developer; this may be an issue of scale rather than fashion. I note here that interview Subject F, an individual developer, uses only a text outliner and a flowchart drawing package in addition to the required tools of a text editor and language compilers and libraries, although he develops systems of some considerable size. The interviewee has clearly not caught up with the latest fashion in support tools, but this has not prejudiced the authors' belief in them.

As another example, am I being unfair in citing Laurence Holt (quoted in Fisher, 1994) as an example of fashion dictating CBSD practice, or at least as evidence of a perception that new things are better than older things? Holt is quoted as saying that "Large consultancies are persisting with 1970s practices such as hiring arts graduates for software development and training them in the use of Cobol rather than fourth generation tools." Holt's perception is that Cobol is 'out of date' and fourth generation tools are the way forward, but, without considering the arguments implicit in that viewpoint, it is instructive to note the reference to "1970s practices". It seems that, to Holt, the tools available improve with time. If this view is taken without careful evaluation of each new tool, it will inevitably become a belief that newer tools are better simply because they *are* newer. This is a fashion-driven approach, in which the latest is *ipso facto* the best.

Finally, I cite a counter-example to the suggestion that following fashion is universal. Blum (1993, p 126) opines that "newness by itself is not a virtue", noting

that many "new paradigms" for views of the CBSD process are in fact re-expressions of existing ones.[61]

Fashion is not always followed, but the costs of not doing so can be considerable in terms of credibility in the community and success in the marketplace. For instance, the retention of a waterfall-style, rigid, feedback-limited process model in SSADM is criticised, since that model is out of favour despite it having some advantages in terms of project management.

6.4.4 Aesthetics in the Externals of a System

It is noticeable from the investigations performed for this thesis that positive elements of the external aesthetic aspects have *not* been considered as a major quality determinant either by the theorists, apart, possibly, from Licker (1987), or by practitioners. It appears that computer-based systems are viewed by most of the investigative subjects as being evaluated, sold or considered on the basis of functionality and price, not aesthetic criteria.

However, there is evidence that this aspect of a computer-based system is considered to be important by developers and users of end user environments; consider the configurability of Microsoft Windows[62] and X screen colours, wallpaper and so on. Such developers see these aspects as points which will encourage people to use their systems more. An equivalent user interface designed purely to enhance functionality on ergonomic grounds would automatically preselect that colour combination which experiment has determined to be the 'best', so there is undoubtedly more to allowing users choice than a desire to maximise usability.

There are artists who use computers as part of their working environment. There are also computer people who use their machines to generate works of art, either using their own talents or by use of algorithmic mechanisms such as fractal programs. It is therefore not the case that computer-based systems and pure aesthetics are mutually incompatible. In addition, aesthetics are considered in the internals of computer systems, in which case they are related in the DM of CBSD to minimisation by the term 'elegance', which carries a semantic burden of aesthetic acceptability.

---

61 See Chapter 8 for a discussion of Blum's work and its relevance to that presented here.

62 Trademarks acknowledged.

It is interesting to consider whether the eventual users of the computer-based systems, designed by such as the interviewees and authors considered above, feel that aesthetics are of so little importance. One might contrast the way in which office furniture is sold, on aesthetics as well as functional benefits, or more significantly the unquestioned acceptance by architects of the importance of aesthetics in the work they design. In looking for reasons for the failure of computer-based systems developers to include the aesthetics of the products they produce in their DM, it can be asked whether the attitude that function is much more important than appearance arises from some degree of underlying thinking of 'users as objects'.

6.5 Summary

In this chapter, I have used the results of the investigations and my earlier theorising on the Kuhn-based model to:

- explain the current divided state of CBSD in terms of schools based on differing DMs; and
- extend the Kuhn-based model of CBSD to consider such aspects as the paucity of symbolic generalisations and interactions between DM elements.

I have also discussed three DM elements, including a detailed exposition on the inductive nature of CBSD.

In the next chapter, I will bring the work on the Kuhn-based theory together by describing a means by which the complete DM underlying a CBSD activity can be conceptualised, based on a list of DM elements such as that provided in Appendix 2 to this thesis.

7 An Algorithmic Model of the Generation of a Disciplinary Matrix

> "Haec autem ita fieri debent, ut habeatur ratio firmitatis, utilitatis, venustatis"
>
> "These things, moreover, should be so performed that a balance is obtained between stability, usefulness and beauty"
>
> (Vitruvius, 1931, p 34; my translation).

7.1 Introduction

As we have seen, CBSD as a discipline is divided into schools of opinion, each of which is based on a differing DM. Within CBSD, there is the potential for disagreement on a variety of issues, each of which generates one or more differences in the elements in the schools' DMs. As a result, the DM structure for CBSD and its component schools is more complex than a sequential list of elements alone can show. This chapter therefore describes, in the form of an algorithm, a model of how the DM which affects a single example of the work of an individual CBSD practitioner might be conceptualised as being generated from a list of DM elements such as that in Appendix 2 to this thesis.

The model is intended to result in the reader having a mental picture of the way in which the DM is made up of a series of interacting beliefs and models in the mind of the computer-based systems developer, rather than being a description of how such a DM actually comes into being in the mind of that individual. There is no suggestion that the mechanism described is related in any way to the mental mechanisms by which the DM actually arises in a developer's mind. It might rather be said that a mental model built in the way in which it is described here from the relevant list of elements should have the same effect on the work of a CBSD practitioner as the real DM actually does. What is described here is thus a model for the conceptualisation of a DM rather than a prescription for how to build one.

The model is presented as a theory building on existing work; suggestions for its testing and use are described at the end of this chapter. It has been written in an informal notation using structured English, and comprises a series of procedures and a number of data structures. Explanatory notes follow the model itself.

7.2 An Outline of the Model

7.2.1 Main Features of the Model

The main features of the Algorithmic Model are:

- the DM elements themselves;
- the sources of these elements – where they arise from, *viz.* the discipline of CBSD, the specific method and tools employed, the system-specific context and influences, the person him- or herself, the organisation for which he or she works,[63] and society in general;
- the ways in which these elements are brought together, in particular the observations that the adoption of some DM elements may imply the adoption of others, and that there is a need to trade off between elements;[64] and
- the way in which this trading off process might of itself result in changes to the weightings assigned to the different elements.

At each stage, the DM elements are weighted according to their perceived relevance and/or importance to the current situation, and these weighted elements are reconciled to produce as much internal consistency as possible.

7.2.2 Sources of Disciplinary Matrix Elements

The CBSD method referred to above is that which is actually *used* for an example of CBSD work, after any tailoring or adaptation phases have been performed. That method comprises the process model, tools and techniques which are actually to be *used* to develop the computer-based system, rather than those which are described in the theoretical exposition of the method. That the DM derives from the method as actually used rather than the method as originally described by its authors is supported by the DM element 'When all else fails, read the manual', which is itself derived from the propensity of computer-based systems developers to use what they believe to be the correct techniques rather than checking in the

63 Support for organisational influences comes from the field of Information Systems, for example van Gigch and Pipino (1986, p 81).

64 This need to trade off is reflected in the DM element list in Appendix 2 as 'A compromise must be made between conflicting priorities during the systems development process'

manual at each stage. This drift of practice from theory might be reduced in some cases by, for instance, the use of computer-based tools which rigidly enforce the rules of the method or some or all of its component tools and techniques.

The DM of the method is in turn derived from DMs for those techniques, notations, process models and support tools. The last of these categories covers CASE tools, workbenches and so on; the DMs for these are currently not assumed to be necessarily identical to those of the methods in which they are incorporated.

Beliefs and models drawn from the wider context of the discipline of CBSD, and from society in general, also influence each instance of CBSD practice, and therefore need to be included in the final DM. Consider, for example, the views of society in Scandinavia on the introduction of new technology and their influence on the mechanisms required to introduce new systems resulting in a recognisable 'Scandinavian School' of CBSD, as evidenced by the quotation cited previously from Groenbaek *et al.* (1993).

In deriving the DMs of individual computer-based systems developers, the elements relating to *that individual's* beliefs, models and so on are also relevant.[65] Since these are drawn from personal experience or beliefs rather than from any common culture, such DM elements are less generally applicable than is the case for the DM elements drawn from the sources mentioned above.

7.2.3 Types of DM Elements

Each element of the DM is described as being either common or dividing, as previously described. It can also be single-valued (it is either believed in or unavoidable, or it is not), or there may be more than one possible value. Multi-valued elements are also described as either:

- continuous – a belief or model for which differing values can be held simultaneously in differing proportions; or
- non-continuous – the element can take one or more of a number of discrete values.

---

65 Gasson believes that the personal aspects of the belief system are more influential than the method-based ones (S. Gasson, personal communication, 1994).

Examples of continuous and non-continuous elements are described later in this chapter. Since they can take different values, they are by definition dividing rather than common elements.

7.3 The Algorithms

In this section, I describe the algorithms for modelling a CBSD practitioner's DM from the DM elements of CBSD.

7.3.1 The DM of a Tool, Technique, etc.

The algorithm for building the DM for a technique, notation, process model, support tool or computer-based systems developer outside the context of a particular development situation is:

```
FOR EACH    technique, notation, process model, support tool and
            computer-based systems developer

      FOR EACH    element common to all computer-based systems
                  developers

            TAKE  element

      END FOR


      FOR EACH    element which divides the discipline into schools,

            IF    element IS relevant

                  IF    element IS 'non-continuous'

                        TAKE  one or more of the possible values

                  ELSE  /* (element is 'continuous') */

                        TAKE  a belief formed of variable proportions
                              of the different possible values

                  END IF

            END IF

      END FOR

END FOR


RESOLVE     interactions between the elements taken to form a combined
            DM, weighting each of the elements with regard to its
```

```
                importance in the context of the current DM as a whole (as
                far as possible); this may in turn result in the addition
                of new elements and/or increases or reductions in the
                significance of existing elements, or in the complete
                elimination of some existing elements.
```

7.3.2 The DM of a Method

The algorithm for building the DM for a method outside the context of a particular development situation is:

```
TAKE        the DMs for the techniques, notations and process models
            forming the method

RESOLVE     interactions between the elements taken, weighting each of
            the elements with regard to its importance in the context
            of the current DM as a whole (as far as possible); this
            may result in the addition of new elements and/or
            increases or reductions in the significance of existing
            elements, or in the complete elimination of some existing
            elements
```

7.3.3 The DMs of a Project and its Component Activities

The DMs for specific development circumstances are presented here. Note that individual, organisational and society-based factors are added in to the predefined DMs due to the method and its components.

The Project Level

Each project can have its own influences on the way in which the work for it is performed. This influence can be considered to derive as follows:

```
FOR EACH    computer-based systems developer X project

    TAKE        the DMs for the society, organisation, project
                context, discipline, computer-based systems
                developer (the personal DM), his or her role, method
                (as derived above) and support tools employed

    RESOLVE     interactions between the elements taken, weighting
                each of the elements with regard to its importance
                in the context of the constraints and contexts of
```

```
                the project[66] and of the current DM as a whole (as
                far as possible); this may result in the addition of
                new elements and/or increases or reductions in the
                significance of existing elements, or in the
                complete elimination of some existing elements

END FOR
```

The Activity Level

Within each project, there are a number of activities to be performed. The way in which DM elements from CBSD and elsewhere can affect each of these is represented by:

```
FOR EACH    computer-based systems developer X activity

      TAKE        the DMs for the society, organisation, project
                  context, activity context, discipline, computer-
                  based systems developer, his or her role, method and
                  support tools employed

      RESOLVE     interactions between the elements taken, weighting
                  each of the elements with regard to its importance
                  in the context of the activity and of the DM as a
                  whole (as far as possible); this may result in the
                  addition of new elements and/or increases or
                  reductions in the significance of existing elements,
                  or in the complete elimination of some existing
                  elements

END FOR
```

7.4 The Data Structures

This section documents the data structures which are needed to support the procedures described above. The data held for each element includes any other elements which it entails, together with weightings to indicate the strength of these connections.

---

66 An example of this weighting process causing the complete elimination of a common element from the final DM can be found in interview subject C's elimination of 'the system as a minimalist construct' from the DM underlying academic proof-of-concept CBSD work.

7.4.1 DM Elements Common to All Computer-based Systems Developers

The information required for a DM element which is defined as common to all CBSD activities is as follows:

```
Element:    the definition and name of the element

    For each other element which this element either implies or
    denies:

    Element:    the element's name

    Weighting:  positive or negative; a base value, derived from the
                'normal' type of interactions with the element being
                defined, and which may be modified due to the other
                influences which cause the weightings used in the
                resolutions of interactions to arise
```

7.4.2 DM Elements Dividing Computer-based Systems Developers into Schools

There are two possible data structures for elements dividing CBSD practitioners into schools, depending on whether they are continuous or non-continuous as defined above.

Continuous

These elements have multiple values, the result in any particular case being a combined value formed of the union of weightings (from 0 upwards) of any or all of the different values.

```
Element:    the general definition of the belief

2+ Values:  the 'primitive' values, the two or more extremes of belief
            between which any position may be taken

    For each other element which this element either implies or
    denies:

    Element:    the element's name

    Weighting:  positive or negative; a base value, derived from the
                'normal' type of interactions with the element being
                defined, and which may be modified due to the other
                influences which cause the weightings used in the
                resolutions of interactions to arise
```

Non-continuous

These elements have multiple values, the result in any particular case being one or more of the discrete values given.

```
Element:     the general definition of the belief

2+ Values:  the possible specific beliefs which meet the general
            definition

      For each other element which this element either implies or
      denies:

      Element:     the element's name

      Weighting:   positive or negative; a base value, derived from the
                   'normal' type of interactions with the element being
                   defined, and which may be modified due to the other
                   influences which cause the weightings used in the
                   resolutions of interactions to arise
```

7.4.3 Sets of Elements Due to Methods, Techniques, Tools, etc.

A data structure is required to describe each DM element, and the weighting given to it before outside interactions are taken into account, which is implicit or explicit in a CBSD tool, technique etc. Given this information, a base DM can be built for a method by referring to the tool etc. for its explicit or implicit DM elements, and then weighting each of these elements according to the importance of the technique etc. within the method.

Techniques, Tools, Notations, Process Models, Support Tools and Computer-based Systems Developers

For each DM element to be included in the DM of the technique, etc.:

```
Element:     the element's name

Weighting:   a value related to the degree of influence which the
             element has on the technique, etc.
```

Methods

For each tool, technique, notation, process model and support tool to be included in the method:

```
Member:     each technique, notation, process model, support tool

Weighting:  a value related to the degree of influence which the
            technique, etc. has on the method
```

7.5 Notes on the Model

I note here various matters related to the model and its interpretation.

7.5.1 General Notes

7.5.1.1 Algorithms and Data Structures

- The order in which the elements are selected is not relevant to the final result, since the comparison of elements does not occur until all which may interact at each stage are selected.
- the 'taking' of one element in the algorithm from the list of DM elements may sometimes imply that one or more other elements may have to be included or excluded from the DM, as reflected in the other elements explicit or implicit in it. This is true for *all* 'take' operations.
- As indicated in the descriptions of the processes of resolving interactions between elements, some aspects of the weightings to be ascribed to the individual elements when resolving interactions are dependent on other DM elements present in that resolution. These weightings reflect a quantification of the idea of interactions between DM elements, which is considered qualitatively elsewhere in this thesis.

7.5.1.2 The List of DM Elements

- Other than the division into common and dividing elements, and the occurrence in the latter of continuous and non-continuous elements, no comment is made for the Algorithmic Model as to the organisation, if any, of the list of DM elements from which each CBSD technique etc. draws the content of its DM, nor about the order of selection of the elements from that list.
- The only sources of DM elements which have been included in the investigations described in this thesis are those related to CBSD and its component schools. Other sources of influences on the computer-based

systems developer, such as society, the organisation and personal beliefs, are outside the scope of this thesis, and have not been considered further here.

7.5.1.3 Notes on the Underlying Theory

- Kuhn states that the DM of a discipline defines:
  - the problems and solutions acceptable to that discipline; and
  - the members of the community in that discipline, by reason of their adherence to that paradigm.

  As stated previously, Kuhn has been criticised for the potential circularity of the definitions of DMs and communities.

  In CBSD, adherence to the common elements of the DM defines membership of the community of computer-based systems developers. This algorithmic approach assumes, however, that the DMs of all techniques, notations, etc. also reflect the base DM of the computer-based systems developer, which in turn reflects the further assumption that method and technique developers combine with practising computer-based systems developers to form a larger global CBSD community.

- It may be that the DMs to be combined in a resolution process cannot be satisfactorily resolved. Locating examples of this is a matter for research beyond the scope of the thesis, as are an examinations of the effects of such failures. However, it seems reasonable to suggest that an inability completely to resolve the DMs can occur, resulting in a method which it may be found difficult to apply successfully.

- Is the complete elimination possible of *any* element of the DM of CBSD during the algorithm? Apparently not in some cases. For instance, the Principle of Uniformity of Systems is logically required to be a part of the DM under which CBSD activities take place. Its influence is impossible to eliminate from the DM of a CBSD process.

  I suggest that it is possible for such an element to be eliminated from the explicit belief system which guides the computer-based systems developer, and will therefore be ignored by him or her. What *cannot* be done is to remove its implicit influence from the course of that CBSD activity.

- An attribute of a method which some regard as useful, that of the 'coherence' of the beliefs underlying it,[67] might be defined as the smooth completion of the operation of resolving interactions between elements of the DM.

7.5.2 Examples of Non-continuous and Continuous Dividing Elements

I present here examples of the two types of dividing DM elements, non-continuous and continuous DM, introduced previously in this chapter.

7.5.2.1 A Non-continuous Element

An example of a *non-continuous* element of the DM of CBSD is the belief that a method can be optimised by its developers in terms of the type of systems for which it is intended for use.

For example, MASCOT (Simpson, 1986) is designed for real-time systems, typically in a defence environment. Other methods, for example, OMT (Rumbaugh *et al.*, 1991) claim a general usability. The values for this aspect are the different domains for which computer systems can be developed, and any or all can be the values selected for a particular method.

7.5.2.2 Continuous Elements

An example of a *continuous* DM element is 'People interacting with a system can be treated as objects, and reduced to (impersonal) roles', which might vary from very low or nil (for example ETHICS, as summarised in Mumford, 1993) to completely (such as Structured Design (Constantine and Yourdon, 1979); see Bansler and Boedker, 1993). The characteristic identifying a continuous element is that the value can vary between two or more possible extremes. The mechanistic treatment of the user is a two-pole example.

Another example of a continuous element is that of the underlying *computational model* embodied in notations and process models. The possible values for this include:

---

67 "...a good information system is realised by making specifications using a coherent set of models at several levels of abstraction." (Essink, 1983, p 55, abstract)

- *procedural* – emphasised in, for example, process models which emphasise project management and the completion of clearly-identified activity milestones;
- *data* – emphasised in such methods as Structured Analysis/Structured Design, and process models which emphasise the products of the stages of development rather than the stages themselves;

  and
- *object-oriented* – emphasised in methods such as OMT (Rumbaugh *et al.*, 1991).

Whilst the computational model might be thought of as a non-continuous element, in that the primitive computational models which might be identified are different, the computational model used for an actual technique, etc. is in fact part of a continuum. Consider, for example, JSD (Jackson, 1983), which can be considered as mixed data-plus-object. The computational model of a notation or process model is thus a multi-pole continuum rather than a continuum with only two extremes, with the number of possible extremes equal to the number of primitive computational models which can be identified.

7.5.3 Testing and Using the Model

The Algorithmic Model as presented here is, in scientific terms, a theory which has not yet been tested experimentally. Before it can be used for further theory-building, it will therefore be necessary to test it by investigative or experimental techniques.

Such techniques might include:

- comparative examinations of existing computer-based systems developed under what are believed to be different DMs; and
- setting up hypotheses concerning specific DM elements either alone or in combination, and testing them using appropriately-selected subjects.

Should the model be shown by these means to be valid, it might then be used for:

- setting up experiments into the effects of DM elements singly or in combination, and interpreting the results of these experiments; and
- modelling DMs and their expected interactions by implementing it as a computer-based system, using the DM as then known as a database, which would have the potential to assist in CBSD project planning.

In the first instance, a simplification might be made by ignoring some or all of the second-order interactions between elements when resolving interactions. This would result in a considerable reduction in the amount of data to be determined and stored in the list of elements. Were this simplification to be adopted, it would be necessary at a fairly early stage of the work to investigate the interactions between elements to determine which are significant and which can be ignored due to their small effect. An example of an effect which should not be ignored, due to its potential for a significant effect on CBSD practice, is the influence of the waterfall process model on novelty in design solutions described previously in this thesis.

7.6 Summary

This chapter has described a conceptualisation of the DM of CBSD, intended to give its readers a view of the way in which the influences of the mechanisms used by the computer-based systems developer on the products of that development arise.

The elements making up the DM of each technique, notation and process model are merged to produce a DM for each, which is a particular influence during the activities using it. The DMs for the techniques and notations making up the method are merged again to form the DM for the method, which is a general influence on all developments using it. This method-derived DM is further merged with the DMs for the individual computer-based systems developer, any other development support tools used and influences from the organisational context, to give the DM which guides the computer-based systems developer in his or her practical work.

I suggest that this model can be tested for validity by comparative examinations of existing work or controlled experiments. After this, it may be possible to use the

model as the basis for further experiments, or to implement it as a computer-based system.

8 Related Work

8.1 Introduction

In my examination of the literature of CBSD, I have been unable to find any work which is directly parallel to mine, i.e. using Kuhn's work as the theoretical basis for a search for a widely-based DM of CBSD, formed of many elements, to support a Kuhnian pre-science school-based model of CBSD theory, and a normal science view of method-based CBSD practice.

I have however noted a number of publications which have some relevance to this work. These provide general support for my approach, or allow the identification of elements of the DM of CBSD to be made, through the explicit statements or implicit assumptions of the authors.

In this chapter I describe some of this related work. Where the conclusions of this review have resulted in the identification of specific elements of the DM, this is reflected in references in the list of elements in Appendix 2 to this thesis.

I have divided the relevant research of other researchers into groups, describing for each group the work in question and then commenting on it. The taxonomy which I have employed for this purpose separates those who have:

- employed Kuhn's work as the basis for the consideration of CBSD or one of its associated or sub-disciplines;
- provided general support for the ideas behind the Kuhn-based model described in this thesis;
- identified elements of the DM of CBSD, either as an end in itself or as the basis for taxonomies of CBSD theory or practice; and
- identified or demanded 'paradigm shifts' in CBSD.

In addition, comments by Green (1989) are quoted to support the apparent complication of the DM and the interactions between its elements, and the nature of the term 'paradigm' is considered. Finally, research is summarised which may of use in extending the work presented in this thesis.

My allocation of each publication to one of these categories does not necessarily accord with the authors' apparent perceptions. It is based on my view of how their work relates to mine, and on placing together work which can be commented on in the same way. I have discussed work which falls into more than one category, such as that of Blum (1993, 1994, 1995), under each relevant heading.

8.2 Direct Users of Kuhn

8.2.1 Introduction

In this section, I describe published research which can be applied to support the application of Kuhn's work to describe CBSD. I derive this support from those who have either employed the broad thrust of Kuhn's work in considering CBSD or an associated discipline, or recognised the influence of the DM on the practice of a discipline.

8.2.2 Banville and Landry

Banville and Landry (1989), writing in a computing journal, seek to provide a theoretically sound basis for research into Management Information Systems ("MIS"). To this end, they describe Kuhn's work, both as originally described (Kuhn, 1962) and as later expounded (Kuhn, 1970), and some of the criticisms made of it.

They then reject Kuhn's work as a basis for examining MIS, on the grounds that:

- it is difficult to identify the community of MIS researchers (*ibid.*, p 49), since such efforts seem to result in a 'paradigm' "that could include the different approaches currently found in the MIS field." (*ibid.*, pp 49–50) – in short there was, at the time Banville and Landry wrote, no unifying DM for MIS;
- the parallel with the perfect Kuhnian 'science' of physics is inappropriate (*ibid.*, p 50); and
- Kuhn finds it difficult to define scientific 'progress' in his model (*ibid.*, p 51), and that advances sought by establishing a paradigm for MIS might not therefore accrue.

8.2.3 Checkland

Checkland (1990) provides support at the most general level for the application of Kuhn's work to CBSD.

The Soft Systems Methodology ("SSM") is designed to be applicable to human activity systems (*ibid.*, p.149), i.e. systems in which humans combine and co-ordinate their activities for some purpose. CBSD activities, being such systems, can legitimately be examined by reference to Checkland's work.

In a section entitled "The importance of *Weltanschauung*", Checkland (*ibid.*, p 215) considers the way in which the *Weltanschauung* ("W") – "the (unquestioned) image or model of the world" (*ibid.*, p 319) – dominates our thinking. To Checkland, the W is a key aspect of understanding any human activity system. For him, a statement about such a system is incomplete unless it is accompanied by the W associated with that statement (p 220). This importance of the W can be applied transitively to the paradigm to which Checkland has referred.

8.2.4 Farhoomand

Like Banville and Landry, Farhoomand (1987) implicitly assumes that MIS is a science. He applies Kuhn's model as a basis for a survey of research trends in that discipline. He notes that shared commitments can be identified in "the specific areas of MIS" (*ibid.*, p 50) rather than in MIS as a whole. These sub-areas can be identified by, for instance, allegiance to top-down or bottom-up approaches.

He suggests that this lack of a unifying DM might be due to the influence of values from other disciplines in a new area (MIS) which has not yet had time to build its own research traditions. However, he also believes that recent developments, such as the emergence of a common MIS curriculum and agreement on the boundaries and goals of MIS, point to a then-forthcoming coalescence around a single DM.

8.2.5 Critique and Comments

I consider below each of the above applications of Kuhn's work, in relation to the work described in this thesis. Of the three examples, I note that those of Banville and Landry and of Farhoomand are each based on perceptions that the disciplines which they are studying are sciences. Checkland's work is intended to be more generally applicable to human activity systems.

I take issue with Banville and Landry on each of their grounds for rejection of Kuhn's work as a parallel. Their difficulty in finding the 'paradigm' of MIS seems to arise from their desire to find a *unifying* paradigm to establish MIS as a science,[68] rather than to recognise the possibility that MIS might be in a state equivalent to that of a Kuhnian pre-science, with competing schools arguing over the paradigmatic structure of the discipline. I gladly accept that CBSD is not a science after the model of physics, but suggest that this is a reason for viewing the use of Kuhn's model as an issue of analogy rather than identity, exploring the differences and adapting Kuhn's work to suit. This also disposes of any worries concerning the nature of progress in CBSD, since a definition of progress in a science may be very different from that for a design discipline; once there is no assertion that MIS or CBSD is a science, then progress in those disciplines may indeed differ from that of physics. Finally, I have resolved the difficulty of identifying the CBSD community by reference to a definition of 'computer-based system', and thus to the community of those people who develop such systems and the tools to support that development activity.

Farhoomand neither provides a critique of the application of Kuhn's work to MIS, nor does he enlarge on the effects on the significance of his recognition of sub-groups of MIS for Kuhn's model. Like Banville and Landry, his approach is grounded in the axiom that MIS is a science, which I contrast with my identification of the parallel discipline of CBSD as a Kuhnian pre-science.

In addition, both Banville and Landry, and Farhoomand, have studied the theorists of the disciplines which they are studying, and not the practitioners. They examine the theoretical literature, rather than the practice. I have from the outset intended my work to cover both the theory and the practice of the discipline which I am examining.

Checkland, in contrast to these two examples, has considered his application of Kuhn's work in the context of the development of a generally-applicable inquiry mechanism. His identification of SSM's *Weltanschauung* with Kuhn's paradigm gives a context within which to place the DM of CBSD, even if it is not clear which of the meanings of 'paradigm' in the first edition of Kuhn's work is being used.

---

68 Banville and Landry (1989, p 48) refer to the "pervasiveness of some of the most active members of the MIS field of the interest for the actual state and future of MIS *as a scientific field*." (my emphasis)

The identification of paradigm with W also provides a possible key to future work which might explore generic CBSD activities in the form of an SSM model. This is suggested as future work in the concluding chapter of this thesis.

8.3 General Support for the Kuhn-based Model

8.3.1 Introduction

Some researchers have made general statements which lend support to the key underlying assumptions of the Kuhn-based model, i.e. that a belief system exists in CBSD and has an effect on the products of this discipline.

Examples of these general statements are cited here.

8.3.2 Constantine

In a paper examining ways of organising CBSD teams, Constantine (1993, p 35) notes the existence of a paradigm, "the model and its incorporated assumptions that guide or inform the organization and operation of a group" of people developing a computer-based system. He then examines the differing ways of organising CBSD teams. He cites Kuhn as a source of this definition, but makes no reference to Kuhn's models of which it forms an aspect.

8.3.3 Dasgupta

Dasgupta (1991, p 141) states that in his view:

> "...the concept of the Kuhnian paradigm plays a role in the design disciplines that is very similar to – indeed is indistinguishable from – its role in the natural sciences".

8.3.4 Episkopou and Wood-Harper

Episkopou and Wood-Harper's (1986, p 222) view of how the 'approaches' to systems analysis identified by Wood-Harper and Fitzgerald (1982), which will be considered as a taxonomy of CBSD below, affect the way in which CBSD activities proceed is that:

> "Once the choice [of approach] has been made, ... the analyst has put on his/her chosen 'glasses' with which to view the problem situation."

8.3.5 Gasson

Gasson (1994, p 140) notes her belief that "A computer system has, embodied in it, all of the assumptions and perceptions of its designer."

8.3.6 Hirschheim and Klein

Hirschheim and Klein (1989) make a series of statements which lend support to the Kuhn-based model. They note that "there are ... alternatives [to the orthodox CBSD approach in 1989] that are based on fundamentally different sets of assumptions." (*ibid.*, p 1199). These assumptions concern the nature of reality, and are "explicitly or implicitly made in adopting a particular development approach" (*ibid.*, p 1199).

Hirschheim and Klein assert that "important social consequences result from adopting a particular systems development approach." (*ibid.*, p 1199). They later note that differences in approach can result in differences in the system delivered (*ibid.*, p 1121).

8.3.7 Petre

In examining the results of an investigation into the mental processes of declarative programmers, Petre (1989, p 60 *et seq.*) notes that some implementation features leak through into the design process (*ibid*, p 64).

This implies that some elements of a personal DM affect the ways in which a tool will be used.

8.3.8 Comments

Each of the above statements provides support for the work presented in this thesis.

Constantine's comment suggests that he visualises some form of DM which has an effect on the practice and products of CBSD. Dasgupta concurs in this. Episkopou and Wood-Harper believe that the approach adopted acts as a filter on the analyst's perception of the problem. The view of Gasson, and of Hirschheim and Klein, that the underlying assumptions of the designer affect a system's design,

implies that a link can be made between the underlying assumptions of a CBSD method, tool or technique incorporating a set of assumptions such as they describe, and the product of a CBSD mechanism. This effect of aspects of an underlying belief system on CBSD is the rationale of the Kuhn-based model. All of these comments are thus supportive of the idea that the DM has an effect on the practice and products of CBSD.

Petre's observation implies that the mechanisms of a method, in particular those implicit and/or explicit in a design notation, will be affected in use by the prior experience of its users. This is as would be expected from the Algorithmic conceptualisation of the DM of CBSD presented in Chapter 7 above.

8.4 Identifying and Using Elements of the Disciplinary Matrix

8.4.1 Introduction

Some CBSD researchers have characterised underlying aspects of CBSD, which they have often identified as 'paradigms', in some cases citing Kuhn is support of this contention. I suggest that, in general, these paradigms are each formed of one or more elements of the DM of CBSD. Some of them may be paradigmatic in Kuhn's sense of being exemplars of good practice.[69] The results of this research have generally been presented by their authors either as conclusions in themselves, or as taxonomies for subdividing CBSD theory and/or practice into what I have identified as analogous to Kuhn's pre-science schools.

Having given examples of these lines of research, I comment on the results obtained from both areas of research in the context of the work described in this thesis.

8.4.2 Identifying Elements of the Disciplinary Matrix

8.4.2.1 Introduction

In this section, I describe some work which I suggest has resulted in the identification of specific elements of the DM of CBSD.

---

69 Or bad practice: see Junk and Oman's (1992) criticism of the code-and-fix systems development paradigm identified by Boehm *et al.* (1984).

8.4.2.2 Bansler and Boedker

In the course of a critique of Structured Analysis ("SA"; Yourdon, 1989) in an organisational context, Bansler and Boedker (1993) describe the "fundamental assumptions" of that method. They identify these as being that:

- "*people are made into objects*, simply perceived as 'system components'" (*ibid.*, p 169, authors' emphasis) in a model of the organisation as machine; and
- the design process is a *problem solving* activity (*ibid.*, p 172).

They identify these as *"the* implicit assumptions" (*ibid.*, p 165, abstract; my emphasis) of SA.

8.4.2.3 Dahlbom and Mathiassen

Dahlbom and Mathiassen (1993, p 86 *et seq.*) have recently proposed a model of the current state of CBSD based on Kuhn's work. They view the majority of the CBSD community as being united by a model of software development based on "...construction, mathematical problem solving by stepwise refinement" (*ibid.*, p 86). This model "defines the discipline of systems development for the majority of practitioners and researchers", and "has become a *paradigm* in Thomas Kuhn's sense, uniting the community, giving it a common identity." (*ibid.*, p 86, authors' emphasis)

Dahlbom and Mathiassen see the problem solving mechanism used in the 8 Queens problem – well-defined, 'hard', abstract, hierarchical, decomposable – as being sufficiently general for them to regard it as being an exemplar of CBSD practice. To them, it expresses both the strengths and the limitations of "one of the most powerful paradigms in our discipline." (*ibid.*, p 86).

8.4.2.4 Dasgupta

In the context of *design paradigms*, Dasgupta considers some of the content of a Kuhnian paradigm for a design discipline, such as the design aspects of computer science. He describes a design paradigm as "...an abstract prescriptive model of the design process that ... serves as a useful schema for constructing practical design methods, procedures, and tools for conducting design." (Dasgupta, 1991, p

142). "A design paradigm is to design what a set of heuristics and/or metaphysical models (within a Kuhnian paradigm) is to science. Thus, the concept of a design paradigm is not (even approximately) equivalent to the concept of a Kuhnian paradigm; it is much less than that in the very same sense that beliefs in models is less than (being a part of) Kuhnian paradigms." (Dasgupta, 1991, p 141).

Dasgupta recognises the following paradigms underlying the design process in computer science:

- *Analysis-Synthesis-Evaluation* ("ASE"), which Dasgupta sees as consisting of three separate activities, with connecting feedback loops, and which he notes is a "...widely believed design paradigm in the engineering disciplines" as well as underlying many CBSD methodologies;
- the *Algorithmic* paradigm, which requires the design problem to be precisely defined, i.e. well-structured, of which an example is the machine-based translation and optimisation commonly found in computer language compilers (Dasgupta, 1991, p 324 *et seq.*);
- the *Theory of Plausible Designs* ("TPD"), a paradigm apparently derived or invented by Dasgupta and his colleagues, which sees the design process as the evolutionary refinement of a set of plausible hypotheses (or 'constraints') about the design. Each hypothesis has a state which reflects the designer's belief as to its validity, initially assumed to be 'true', but which can be verified or refuted as part of the design process, based either on further constraints supporting it or on evidence obtained by the designer. The design is never fixed, but can be modified at any time if a constraint's plausibility state changes, for example from 'accepted' to 'refuted'.

The design paradigm forms a prescriptive model, based on one or more members of the design community seeing the design process in his or her field in a particular way, viewing the nature of the design process as of a particular kind, or having some similarity to another activity (Dasgupta, 1991, p 142). It forms a model that members of the design community "...may come to believe in or be committed to. Indeed, a shared belief in, or commitment to, the model may become the reason for an identifiable or distinct community to form. Various 'schools' may become associated with the design paradigm." (*ibid.*, p 142). A design paradigm can support more than one design method, which latter Dasgupta defines as a "...(more or less) explicitly prescribed set of rules ... A paradigm ... may – and in general will

– provide a framework or schema for one or more design methods, just as it may serve as a framework or schema for descriptive and automated tools" (*ibid.*, p.142).

8.4.2.5 Klein and Hirschheim

Klein and Hirschheim (1987), referring specifically to data modelling during the analysis stage of a computer systems development activity, identify two underlying philosophical bases for some of the current techniques – to which they refer as *"the* core assumptions" (*ibid.*, p 8; my emphasis). They identify one as objective and based on a causal model ('objectivist'), and the other as subjective and denying a causal model for human activities ('subjectivist'). The objectivist paradigm underlies entity-based modelling, whilst the subjectivist supports rule-based data modelling mechanisms. Klein and Hirschheim believe that the objectivist position cannot be supported philosophically, and that, as a result, weaknesses are apparent in practice. They cite in support of this the example of the objectivist assumption ignoring the social context of the use of a database, with implications for attitudes to user participation and security.

More recently, Klein and Hirschheim have modified their analysis (Hirschheim and Klein, 1989) to include another two-valued dimension related to beliefs as to the nature of society – order (stability, consensus) vs. conflict (conflict, disintegration, coercion), due to Burrell and Morgan (1979). This, cross-referenced by Burrell and Morgan with an objectivist/subjectivist division, yields the 'four paradigms' in the title of Hirschheim and Klein's paper (1989). The latter use these four paradigms as philosophical positions from which to examine in the broadest terms the possible outcome in systems development scenarios.

8.4.2.6 Liang

Liang (1994) notes Kuhn's statement that a science only advances when it has united under a single paradigm. He therefore provides a theory of information for MIS in the form of a mathematically-based model in a conscious attempt to provide symbolic generalisations for the paradigm of the discipline of MIS (*ibid.*, p 648).

The theory presented is intended to be independent of technology. It represents the fundamental entities of information systems as being data, information, knowledge and wisdom (*ibid.*, p 651). These follow mathematically-based rules,

and sets of definitions, axioms and postulates. The axioms relate natural languages, coding functions, transformation functions and the mapping of perceptions into the human mind. It is postulated that the purpose of information systems is to reduce the entropy of entities, i.e. increase their orderliness (*ibid.*, p 658), and that the purpose of computer-based information systems is to achieve that orderliness by artificial means and to complement the shortcomings of human information processing systems (*ibid.*, p 651).

Liang claims that his model defines a scope for the theory and science of information and provides support for that theory (*ibid.*, p 659). His ultimate goal is to establish MIS and information science as academic disciplines.

8.4.3 Using Elements of the Disciplinary Matrix as Taxonomies

8.4.3.1 Introduction

A number of researchers have identified aspects of CBSD which are common to certain theories and/or practices. They have then used these criteria to divide CBSD into groups, either citing examples of the different categories, or relying on the reader to supply examples of their division of the CBSD community.

Examples of these are described and commented on in this section.

8.4.3.2 Blum

Blum has published a series of papers (including 1993, 1994, 1995) based on a model of the software process which he calls "adaptive design" (1993). He describes a computer-based tool ('TEDIUM')[70] to support this model in practice in developing and maintaining (specifically) information systems, and reports the results of experience in its use.

As part of the theory underlying adaptive design and 'TEDIUM', Blum (1993) has identified two sets of what he terms "paradigms":

- the *orientation* of the software process, towards either a *product* or a solution to a *problem*; and

---

70 "TEDIUM is a registered trademark of Tedious Enterprises, Inc." (Blum, 1993, p 122).

- the *degree of computer tool support*, which varies from the obsolete *standard* (executable code storage only) via *tool-enhanced* (separate design and implementation tools) to Blum's goal of *fully-automated* support in which the support tools can store complete application information and generate an executable solution from this material.

The first of these sets corresponds in my work to a two-valued element of the DM of CBSD. The latter finds expression in the list of elements in Appendix 2 in the form of the *attitudes* of computer-based systems developers to such tools.

Blum's beliefs concerning the need for, or desirability of, paradigm shifts in CBSD are considered below.

8.4.3.3 Boehm et al.

Boehm *et al.* (1984) initially identified three different approaches to developing computer-based systems:

- *building-and-fixing* – coding without specifying, a mechanism which is what is usually referred to informally in CBSD circles as 'hacking';
- *specifying*, which is essentially the 'traditional' analyse-design-implement-rework phased development process; and
- *prototyping*.

They have employed these process models as a taxonomy of CBSD mechanisms, and used the second and third as the basis for an experiment into their comparative effectiveness. This work ultimately resulted in the design of a fourth approach, the *spiral* process model (Boehm, 1988), which employs aspects of both specifying and prototyping paradigms.

8.4.3.4 Gasson

Gasson (1994) describes a "framework to consider the impact of IS [information systems] methods" (*ibid.*, p 141). This framework is based on seven criteria. For

each criterion, a continuum is defined between two extreme positions. The continua and their extreme positions are (*ibid.*, p 143, Fig 2)[71]:

- development priorities – *from* technical optimisation *to* work & social system "design" for socio-technical use (*ibid.*, p 142–143);
- extent of user participation – *from* low *to* high;[72]
- approach to problem investigation – *from* top-down *to* bottom-up;
- modelling approach – *from* function-oriented *to* human work process-oriented;
- control of development processes – *from* formal *to* informal;
- project life-cycle time-scale – *from* long *to* short; and
- project life-cycle model – *from* waterfall process *to* evolutionary development.

Using this framework, it is possible to place a method in the context of the underlying beliefs of its developers, and Gasson does so for different types of method, such as structured vs end-user development methods, and throwaway vs evolutionary prototyping methods (*ibid.*, p 148, Fig 3).

8.4.3.5 Wood-Harper and Fitzgerald; Episkopou and Wood-Harper

Wood-Harper and Fitzgerald (1982) identify six main approaches to systems analysis, based on:

- general systems theory;
- human activity systems, of which they give Checkland's method as an example;
- user participation, such as Mumford's work;
- the 'traditional' (National Computing Centre ("NCC")) approach;
- data analysis, using data as the fundamental building blocks; and
- structured systems analysis.

---

71 Gasson uses the diagrams referred to here in her teaching (S. Gasson, personal communication, 1994)

72 Gasson states that user exclusion has "hidden costs" (*ibid.*, p 143).

They claim that, apart from the general systems theory-based approach, all are used in industry. They then look to the deeper bases of these approaches, dividing them into those based on underlying 'science' (traditional, data analysis, structured) or 'systems' (general systems theory, human activity, participative) paradigms.

The sixfold taxonomy of approaches is used by Episkopou and Wood-Harper (1986) as the basis for a means of selection of an appropriate mechanism for different CBSD situations. In discussing these approaches, Episkopou and Wood-Harper (*ibid.*, p 223) note that they are not mutually exclusive, and can be merged successfully.

8.4.4 Critique and Comments

8.4.4.1 General Comments

There are some aspects of the work of others in identifying elements of the DM which it is useful or interesting to observe:

- those who have developed taxonomies of CBSD theory and/or practice have in fact identified some of the differing schools predicted by the Kuhn-based pre-science model of CBSD;
- the workers cited above have each identified comparatively few DM elements when the list forming Appendix 2 to this thesis is considered. What they have found, taken either individually or together, is not the constellation of group commitments which Kuhn expects; and
- it is possible to identify cases of overlap or identity between some of the elements found or used by the different workers, such as the apparent identity between:
  - Dasgupta's (1991) analyse-synthesis-evaluate design paradigm and Boehm *et al.* (1984)'s specifying approach; and
  - Bansler and Boedker's (1993) identification of SA as a *problem solving* activity, and Blum's (1993) *orientation* of the software process towards a solution to a *problem*;

  these may reflect elements of the DM of CBSD which are either widely held, in the former case, or widely criticised, as in the latter.

8.4.4.2 Specific Similarities and Differences

It is possible to identify specific areas of agreement between my work and that cited above. Similarities include shared assumptions that:

- a philosophical basis exists for each method, technique, tool, etc. which can be identified separately from that method etc.;
- this philosophical basis includes metaphysical assumptions and other accepted but not provable beliefs;
- the philosophical bases are at least to some extent common across more than one method, technique, or example of CBSD practice;
- for some of these underlying beliefs, there can be more than one valid value influencing CBSD activities (Dasgupta 1991; Klein and Hirschheim, 1987), whereas for others there is only acceptance or disagreement (Dahlbom and Mathiassen, 1993);
- for those who have identified more than one DM element, there may be interactions between these elements requiring a trading off process (Gasson, 1994; Klein and Hirschheim, 1987, p 13; Episkopou and Wood-Harper, 1986, p 223); and
- making explicit and examining these philosophical bases is worthwhile, because the practice of CBSD can be improved by understanding and/or changing this underlying basis – consider for example Dahlbom and Mathiassen (1993), Dasgupta (1991) and Liang (1994).

Liang's work in particular merits attention because it uses Kuhn's ideas directly as the rationale for research into aspects of a discipline allied to CBSD. Its focus also shows that the current lack of symbolic generalisations in the discipline's underlying theory is troubling for researchers other than myself.

However, there are also clear differences between my research and much of that cited in this chapter, which include the following:

- most of the other researchers cited above as having examined CBSD or an associated discipline have restricted the scope of their work in one or more of the following ways, by limiting their analysis to:

- one, or a small number, of aspects of the underlying belief system implicit or explicit in a CBSD method, such as Wood-Harper and Fitzgerald's approaches (1982), Klein and Hirschheim's (1987) continuum from subjectivist to objectivist approaches, or Bansler and Boedker's (1993) "fundamental assumptions";[73]
- *method-based* CBSD or well-defined techniques or families of techniques: consider, for example, the elimination of the *building-and-testing* approach from Boehm *et al*'s (1984) experiments. The work presented in this thesis is intended to reflect both method- and non-method-based (informal) CBSD work;
- a restricted set of types of method or technique, such as the work of Klein and Hirschheim (1987) on data modelling;
- a subset of CBSD, such as information systems development (Hirschheim and Klein, 1989; Gasson, 1994) or the design problems of computer science (Dasgupta, 1991);
- a restricted work context of CBSD activity, such as 'organisations' (Bansler and Boedker, 1993); and/or
- some of the stages of the life-cycle of a software product (e.g. Klein and Hirschheim (1987) and Wood-Harper and Fitzgerald (1982), neither of which make reference to the implementation and maintenance phases.

In the work presented in this thesis, I have aimed at a wider scope than the above, in that:

- I have taken as broad a view as possible as to the elements forming the DM of CBSD. The trawling work has been intended to determine as many as possible of the dimensions of that DM, rather than restricting itself to the identification and/or confirmation of a small number of these; and
- in examining CBSD practice, I have included workers using informal practices as well as those using formalised mechanisms, and not restricted the tools used or the area of activity;

---

73 Since these authors identify individual or small numbers of DM elements as the bases of their work, this type of work forms a source of such elements. Their use as such is suggested in the section on future work in the conclusion to this thesis.

- some of the elements or 'paradigms' suggested by other workers, such as the six approaches of Wood-Harper and Fitzgerald (1982) and the 'four paradigms' of Hirschheim and Klein (1989), seem to me to be higher-level beliefs than those listed in Appendix 2 of this thesis. The adoption of one of these might entail a number of elements in my list.

  These higher-level constructs could be of future use as abstractions for grouping the list of DM elements into higher-level sets, and thus aid in the consideration of how both higher-level and lower-level beliefs come into existence;

- I take issue with some of the other workers' identifications of DM elements. In particular, I disagree with some of the details of Gasson's (1994) work. For instance, her implicit assertion that technical and social viewpoints are the only two possible extremes of development priorities must be questioned, the more so in the context of my findings described above regarding the difficulty in defining software 'quality'. I believe that regarding these two extremes as independent values involves a lesser degree of judgment than Gasson's model. Her extreme values in this case also relate only to information systems involving human–computer interaction, which reflects the scope of her area of interest, excluding, for instance, embedded computer-based systems;

- I intend my work to cover both the development of CBSD mechanisms and their application in the real world; some of the cited work (such as Liang, 1994) is directed explicitly to the research community developing tools for CBSD or a related discipline, rather than those who use those tools. However other workers are interested in practice as well as theory, either in developing a theory (Dahlbom and Mathiassen, 1993), testing it (Boehm *et al.*, 1984), applying it in general terms (Episkopou and Wood-Harper, 1986) or developing a product (Blum, 1993, 1994, 1995); and

- some tendencies in others' work have the potential to colour the objective assessment underlying their philosophical explanations, such as:

  - researchers having developed their own CBSD methods and/or frameworks on the basis of their philosophical analyses, in particular 'TEDIUM' (Blum, 1993); and

  - research being presented explicitly in order to support the case for a particular style of development, such as that of Gasson (1994), which suggests strongly that she regards user participation and a socio-

technical approach to CBSD as vital for organisational information systems,[74] and Klein and Hirschheim's (1987) concerns for the social aspects of a computer-based system noted above;

the work presented here is oriented towards as objective an examination of existing CBSD theory and practice as possible, as observed in my own DM presented earlier in this thesis.

8.5 Paradigm Shifts or the Establishment of Paradigms

8.5.1 Introduction

Some CBSD researchers have claimed to identify paradigm shifts in CBSD theory or practice. Others suggest that such a shift is required. In this section, I consider these in the context of the Kuhn-based model of CBSD.

8.5.2 Blum

In the work described above, Blum (1993, 1994, 1995) appears to have changed his views concerning the need for, or the desirability of, a paradigm shift by his readers.

In 1993 (p 117), "The objective of this discussion is to encourage a paradigm shift (in the sense of Kuhn) from our initial perceptions of software development to a design approach better suited to our emerging understanding of the software process." In 1994 (p 215), a paper based on much of the same work "...does not propose a paradigm shift. Rather it seeks to extend the thesis presented by Brooks in his 'No Silver Bullet'." By 1995, in a paper describing in detail the maintenance experience of a system developed using 'TEDIUM' (Blum, 1995, p 25), "A paradigm shift is required, and – as Kuhn has pointed out – that is a difficult and painful process for the practitioners of a normal science. ... We need to change our orientation from one of product creation to one of problem solution."

8.5.3 Connaughton and Dampney

Based on two industrial case studies, and following a short summary of Kuhn's work, Connaughton and Dampney (1994) suggest that CBSD projects have failed

---

74 Top-down and bottom-up views of a system design "should be combined with a perspective of the system as a social work system" (*ibid.*, p 145).

"because major shifts are occurring in the paradigms governing software engineering" (*ibid.*, p 115). They equate the consequences of technical innovation in CBSD on its practice with the effects of a Kuhnian paradigm shift on a science's activities. The viewpoint adopted is that of the non-technical manager who needs to manage a project in which technical aspects may change during its lifetime. They believe that management theories and practices can usefully be imported from general project management, and disagree with the view that CBSD has unique project management problems. The typical CBSD technical expert who is promoted to management seems to be deprecated as a type when compared with the management expert who happens to be managing a CBSD project (*ibid.*, p 118).

Their view is that technical changes need to be allowed for in project management mechanisms, and that these management mechanisms should be considered separately from (faster-moving) technical issues. They conclude that the existence of paradigm shifts in CBSD tools and techniques and their effect on CBSD practice should be recognised as a part of managing CBSD work, and that the introduction of new technical aspects needs to be carefully managed. They conclude (*ibid.*, p 122) that the technical uncertainties in managing CBSD make it like managing research and development, and that flexible management approaches must parallel the computer-based systems developer's happiness with new technologies.

8.5.4 van Gigch and Pipino

Rather than suggesting a requirement for a paradigm shift, van Gigch and Pipino (1986) employ Kuhn's work as the basis for an appeal for the research discipline of Information Systems ("IS") to define, and coalesce around, a single paradigm *(ibid.*, p 71).

They claim that a field cannot innovate or advance until it can break out of the comfortable normal science thinking in which "in a sense" it currently resides (*ibid.*, p 72), and that therefore such a unifying paradigm is required. They also provide criteria for assessing a new paradigm (*ibid.*, p 89), based on those described by Kuhn. Finally, they demand continued flexibility of thought on the part of IS researchers, and an avoidance of locking in to fixed mind-sets.

8.5.5 Critique and Comments

There is no reason why a paradigm shift should not occur in a discipline which has not fully articulated its DM. A significant change in one element, either explicitly formulated or implicitly present, would suffice. However, I have suggested previously that CBSD is advancing, if that is the correct word, more by paradigm *splits* than by paradigm shifts. The theory is moving too quickly for the CBSD community to keep up, and the coalescence around a new DM seems not to occur before another paradigm shift is demanded or claimed.

The community-level paradigm shifts demanded by Blum (1993, 1995) seem to be more a demand that the world catch up with the vision of the author.

The work of Connaughton and Dampney (1994) parallels mine in some ways, and, like mine, has resulted in an identification of the dichotomy between technical and management issues which is indicated by the organisation of the list of DM elements in Appendix 2 of this thesis. The focus of Connaughton and Dampney's work is on high-level management issues, and the detailed technical issues arising from my work have been deliberately ignored by them. Their work supports the 'normal science' view of individual examples of CBSD practice, which might indeed be subject to paradigm shifts at that level. Whether input from each of the individual workers concerned actually allows a complete, invisible revolution to occur at the development team level is unclear. However the idea that the retraining of a complete team in a new CBSD method based on a different DM, and their conversion to the viewpoint embodied in that DM, has similarities to a paradigm shift in a scientific community, has some attraction.

Van Gigch and Pipino's (1986) demand for a single unifying paradigm lends tacit support to the idea that such a paradigm has not yet arrived for IS. I suggest that they have accepted the pre-science state of that discipline, in a similar fashion to my model of the current position of CBSD. In complaining that IS researchers are working in a 'normal science'-like environment whilst urging that a uniting paradigm is needed, van Gigch and Pipino have described a situation analogous to that of a Kuhnian pre-science, in which Kuhn (1977, p 295, footnote 4) has observed the existence of competing paradigms. Given a paradigm within which to act, it is not surprising that individual research activities, even in a divided discipline, should look like normal science. However, being in a pre-science state does not guarantee that a discipline will combine under a single paradigm, as van

Gigch and Pipino suggest. I feel that their demand for a paradigm is a consequence of their desire to see IS as a science, resulting in a belief on their part that such a unifying paradigm must exist. I have stated that I have no opinion as to whether this unification is likely ever to happen to CBSD, and that in my view the use of Kuhn's work to examine CBSD does not imply that CBSD is a Kuhnian 'science'. As a result, and unlike van Gigch and Pipino, I do not have as a goal a search for a single, unifying paradigm for CBSD.

Overall, the literature on claimed or suggested paradigm shifts in theory and/or CBSD practice confirms my view that personal paradigm shifts can occur in individuals in the CBSD community. However, I have found no evidence in this work for any example of an analogue of Kuhn's paradigm shift *viz.* a change in the shared belief system common to *all* practitioners in a discipline, in CBSD as a whole. I suggest that this absence may be due to the fragmented state of the DM of CBSD, mirroring its division into pre-science schools. However, since it might be argued that these schools have their own DMs and might therefore undergo paradigm shifts, have any such revolutions been completed? Two examples of informal CBSD practice from the interviews described previously suggest that the earliest paradigmatic CBSD mechanisms are still in use. These are the use by interview subject F of flowcharts as his main design notation (transcript, p 35), and the 'code-and-test'-like approach of subject B (transcript, p 11) to his research-oriented activities. Remembering that the latter mechanism was so completely deprecated by Boehm *et al.* that they excluded it from their experiments, we may perhaps have here an example of a paradigm shift on the part of a school of theorists not reaching into all corners of practice.

8.6 An Example of Paradigmatic Structure in Computer-based Systems Development

It is interesting to note that some consideration has already been given in an attempt to reduce the influences of a part of the CBSD development environment on the thinking of a software designer to a set of 'dimensions' (Green, 1989).

In the context of comments on implementation languages, Green comments that physicists have reduced all of their units of measurement to combinations of three

orthogonal fundamental dimensions, *viz.* mass, length and time[75]. "Rather cheekily, I suggest that in HCI we attempt to tread the same path." (*ibid.*, p 443)

However, his attempt does not result in a small set of orthogonal dimensions like the basis of physics (*ibid.*, p 444):

> "Cognitive science ... is ... more like plant biology than like physics, in that a whole host of factors (some of them interdependent) affects the growth of plants. Nevertheless our urgent need is for a small number of clear, powerful ideas, which we can pretend are orthogonal."

Whilst my search for the factors making up the DM of CBSD has not resulted in a *small* number of orthogonal dimensions, it has raised my suspicions that Green's analogy of plant biology in terms of the combined influence of a confusion of interacting elements is all too valid, resulting in a DM formed of a large number of inter-related beliefs and models, requiring a trading-off process conceptualised in Chapter 7 above, which stands in stark contrast to the simple, small number of clearly identifiable 'paradigms' suggested by other workers cited above. It may be an important conclusion to be drawn from my work that the dimensions forming the DM of CBSD cannot be considered as orthogonal.

8.7 Thoughts on Defining Paradigms

One feature of those papers cited above which used the term 'paradigm' has been the divergent definitions accorded to the term.

Klein and Hirschheim (1989) define the term paradigm as "The most fundamental set of assumptions adopted by a professional community that allows its members to share similar perceptions and engage in commonly shared practices" (1989, p 1201). They base this on Burrell and Morgan's (1979) definition of the term, which is "meta-theoretical assumptions about the nature of the subject of study" (Hirschheim and Klein, 1989, p 1201, note 4). In the explanatory footnote, Klein and Hirschheim contrast this with Kuhn's definition of the paradigm as *exemplar* rather than that of the *disciplinary matrix*. However I suggest that their 'paradigms' are in fact equivalent to Kuhn's disciplinary matrix.

---

75 Some physicists add a fourth dimension to this; that of charge (R. Winder, personal communication, 1994).

A high-level model underlying a method is described by Wood-Harper and Fitzgerald (1982) as the 'paradigm', using the term "...in the sense identified by Kuhn as a specific way of thinking about problems encompassing a set of achievements which are acknowledged as the foundation of further practice." (Wood-Harper and Fitzgerald, 1982, p 14). This defines a paradigm as a set of exemplars, rather than as a widely-based disciplinary matrix.

These examples show that it is dangerous to rely on a common definition of the term 'paradigm' among different published sources which claim to rely on Kuhn's work. The reason for this problem may be the confusion generated by the use of the term 'paradigm' in the first edition of Kuhn's work (1962). It is in part with this confusion in mind that I have used the term 'disciplinary matrix' ("DM") in this thesis, as reflecting the multidimensional nature of Kuhn's "constellation of group commitments" (1970, p 181).

8.8 Work of Future Relevance

Many of the publications cited in this chapter have been or would be of use in the task of finding elements of the DM of CBSD. To this extent they are typical of the literature which could be used for this purpose, as is suggested in the section on future work in the concluding chapter of this thesis.

In addition to the work considered above, there are other published results or theories which are not of relevance to the work described in this thesis, but which may become relevant and useful as the work on the Kuhn-based model of CBSD is advanced. Examples of these are given here.

Reports of investigations into experience in specific areas of practice, for example Curtis *et al.* (1988), may be a fruitful source of DM elements, as well as on the effects which such beliefs and models have on the practice of CBSD.

Researchers have reported the deeper philosophical assumptions which underlie their CBSD tools, or described philosophical bases for CBSD in general, usually in the context of IS. These might with advantage be examined as part of future extensions to the work presented in this thesis. Such relevant work includes:

- that describing higher-level approaches or paradigms, such as Wood-Harper and Fitzgerald (1982) and Hirschheim and Klein (1989) reviewed above;
- that of Checkland (1990) concerning the Soft Systems Methodology;
- the rationales for the selection of phases and tools for CBSD frameworks such as Multiview (Avison and Wood-Harper, 1993); and
- examinations of CBSD theory and practice from a philosophical standpoint, such as that in Winder *et al.* (1996),

which might allow the correlation of specific sets of the DM elements in Appendix 2 with single higher-level beliefs or models from other sources.

Research such as that of Constantine (1993) into models of organisation of CBSD development teams may also be of use in determining the influences of the roles which CBSD workers adopt in their work, and of the variety of possible interactions both between these roles and with others involved in developing computer-based systems. The DM might be placed in the context of the process of design by reference to such work as that of Curtis (1990).

A further literature-based extension to the work on the Kuhn-based model would be to introduce relevant research from other disciplines. This includes both work from parallel disciplines, such as operational research (see, for instance, Jackson and Keys (1984)) and architecture (a parallel noted by Zachman (1987)), and from disciplines relevant to a 'science base' for CBSD, such as psychology and sociology, to place the DM in its mental and social contexts.

In considering the design of experiments to test the validity or practical effects of DM elements, previous work describing usually student-based experiments such as that of Floyd (1983), Boehm *et al.* (1984) and Junk and Oman (1992) may be of use.

8.9 Summary

In this chapter, I have described and commented on the work of other researchers insofar as it relates to my research described in this thesis.

Whilst there is no work directly parallel to that presented here, I have found that others have used Kuhn's work as a basis for the examination of CBSD or related

disciplines. Others have made statements which lend support to the Kuhn-based model of CBSD. Taxonomists of CBSD and others have identified elements of a DM of CBSD. Demands have been made for changes in the DM of CBSD.

Research by others, some of which is identified in my taxonomy of related work, may also be of use in extending the work in this thesis. I detail the future directions which the research described in this thesis might take in the next chapter, which summarises the conclusions of the work presented in this thesis.

9 Conclusions and Future Work

9.1 Introduction

In this chapter, I summarise the conclusions and results drawn from the work presented in this thesis, in the context of the claims made in the opening chapter, and what has been learned about the usefulness of the Kuhn-based model of CBSD. I also reconsider the critique of CBSD in Chapter 2, and the previously quoted criticisms of Kuhn, in the light of what has been learned.

I finally outline ways in which the research described in this thesis might be extended in the future.

9.2 Conclusions of the Thesis

9.2.1 What Have We Learned?

The work presented in this thesis has:

- demonstrated by argument and/or investigation that:
  - the drawing of an analogy between Kuhn's model of a science and the current state of CBSD is reasonable;
  - the analogy of Kuhn's work portrays CBSD theory as being in a state analogous to Kuhn's pre-science and CBSD practice as being a normal science-like activity;
  - the DM of CBSD, and a structure of common and dividing elements within that DM, exists;
  - a number of specific elements of the DM of CBSD can be identified from current theory and practice;
  - an algorithm-based model can be designed to aid the conceptualisation of how the DM as an entity is related to its set of component elements; and
  - research into the DM might be used to assist in understanding and advancing CBSD;
- raised more speculative considerations concerning:
  - the finer structure of the DM of CBSD;

- the possible existence of other elements of the DM of CBSD not currently supported by investigative evidence;
- the complexity of beliefs or models identified by others as paradigmatic elements of CBSD, but which I regard as being more complex than single DM elements; and
- how the discipline of CBSD, particularly its practice, might benefit from the work presented here.

9.2.2 Confirming Initial Claims

Considering the claims made in the opening chapter of this thesis in the light of what we have learned, it can be observed that the following have been explicitly demonstrated:

- Kuhn's view of science is a reasonable model for CBSD, in that it provides an analogy which explains the current state of CBSD;
- Kuhn's model of science is a useful analogy for CBSD, since it allows an examination of the underlying DM in a way which reflects concerns as to how that DM affects CBSD theory and practice;
- the DM of CBSD exists, which has been shown by exhibiting some of its component elements;
- some of the elements of the DM can be identified by an examination of current theory and practice, demonstrated by example; and
- the elements forming the DM have an effect on the work of the computer-based systems developer – for example, criteria adopted for assessing 'quality' in a computer-based system differ because of different underlying beliefs.

Some questions which remain to be answered in applying Kuhn's model to CBSD include the following:

- are there more symbolic generalisations than have been identified so far?
- how large are the communities involved in the discipline of CBSD?
- can a design discipline in general, and CBSD in particular, coalesce around a unified DM as Kuhn claims for a science?

These questions will need to be probed in the future in order to produce a theory for CBSD, possibly extensible to all design disciplines, which forms a complete parallel to Kuhn's work in the physical sciences.

9.2.3 How Useful is the Kuhn-based Model?

Is the Kuhn-based model a good analytical tool for investigating the current state of CBSD? How has the Kuhn-based model performed as a theory-building tool for examining the current state of CBSD? Is the model useful in developing our understanding, and therefore our practice, of CBSD?

9.2.3.1 As an Analytical and Theory-building Tool

Is the Kuhn-based model a useful analytical tool for examining the phenomena of current theory and practice, or is the work presented here more of a reflection of my own analytical powers than of the abilities of the mechanisms employed?

I suggest that the arguments and investigations described earlier hold the answer to this question. The process of looking for elements of a DM for CBSD was initiated in order to provide evidence concerning the DM's suggested existence and influence. Elements have been found of a core DM uniting computer-based systems developers in a community sharing these beliefs, and other elements dividing this community into schools. To that extent, the use of the Kuhn-based model has been pragmatically successful.

It may be that some of the results obtained have been due to some innate analytical ability or experience in the investigator rather than the underlying theory employed. This suggests that different investigators would have obtained different results using the same techniques on the same subjects of investigation. I believe that this lack of certainty is inevitable in such an early investigation, before the basis of a theory to support true experimentation has been established. The question of repeatability can only be determined by having others replicate my work and determining whether the mechanisms adopted are generally useful.

Two examples of theory-building have emerged from research based on the Kuhn-based model, other than the derivation of the model of CBSD itself. These are:

- the inductive nature of CBSD, as enshrined in the Principle of Uniformity of Systems; and
- a theory of the relationship between the waterfall process model and novelty in CBSD practice.

Both of these are derived directly from the mode of thinking encouraged by looking at a discipline with the theories of Kuhn in mind. The latter is in addition an example of building a testable theory using the Kuhn-based model as a basis.

I contrast the Kuhn-based model as an experiment-building tool with the theories tested by other researchers who have compared CBSD methods and processes, such as Floyd (1983) and Boehm *et al.* (1984). Whereas they have been concerned with the explicit phenomena of CBSD mechanisms, the Kuhn-based model supports an emphasis on the underlying influences of these mechanisms on the practice and results of CBSD activities. I suggest that the experiment-building power of the Kuhn-based model lies in this focusing of the experimenter's mind on the underlying DM rather than the directly visible phenomena, and in the intellectual tools provided to organise the results – the pre-science and normal science views of CBSD theory and practice, and the concept of the DM. Again, the validity of this assertion can only be tested by exercising the theory in practical experiments.

9.2.3.2 As a Practice-Improving Tool

To what extent has the work presented here produced directly implementable results for the CBSD practitioner?

At this early state of development, the adoption of the Kuhn-based model has resulted in one piece of immediately applicable advice for CBSD practitioners. This is for them to examine carefully the inductive assumptions implicit in CBSD activities, both from the viewpoint of their own working practices and to warn others involved or associated with that work. They must note in particular that a problem might arise in a project if a process model without feedback is employed and the inductive assumptions made in designing or selecting CBSD work practices are not justified in practice (Winder and Wernick, 1993).

9.2.4 The Critique of Computer-based Systems Development Revisited

At this stage it is pertinent to ask whether the work on the Kuhn-based model of CBSD is relevant to the criticisms made earlier in this thesis of the current state of the discipline. These criticisms can be summarised as:

- its failure to live up to the name of 'Software Engineering', defining an engineering discipline in the terms presented by Long and Dowell (1989);
- the repeated failures of CBSD projects to meet functionality and time targets;
- its failure to provide a development mechanism or set of mechanisms which have gained the adherence of all developers, particularly the individual or small-scale developers who may employ informal mechanisms;
- its inability to provide a general underlying theory of the development of computer-based systems; and
- the lack of an explicit understanding of the underlying influences on CBSD projects which can be taught to aspiring developers other than by worked examples, which inevitably rely on a limited number of techniques and/or methods and therefore only teach those specifics.

How well does the model presented in this thesis address each of these points?

9.2.4.1 Failure to Define Software Engineering and the Software Engineer

On the basis of the DM of CBSD presented above, can we begin to define what is 'Software Engineering'? I suggest that by a critical examination of the elements forming that DM, and how they interact to produce a set of influences on a computer system's development, we will be able to gain a pragmatic[76] understanding of how CBSD works, and therefore to improve its practice by the initial derivation at the start of a project of a relevant set of elements chosen to influence the development in a particular manner. In addition, since the DM includes information and theories from the underlying science base, the identification of these inputs, and an understanding of how they fit into the development process and interact with other beliefs and models, would in Long and Dowell's (1989) terms advance the state of the discipline towards that of an

---

76 As against one derived from the underlying human science base, such as psychology.

engineering discipline by providing a firmer base for the application of scientific principles.

Further, the work presented here allows us to consider the definition of the term 'software engineer' and what it is to have skill in Software Engineering. One might suggest that, if a computer-based systems developer accepts all of the DM elements defined as being common to all 'software engineers', he or she can be considered to be a 'software engineer'. However, since each element is weighted as part of the process of building a DM, as conceptualised in Chapter 7 above, the questions of 'who is a software engineer?' and 'when does Software Engineering occur?' are not as simple as implied above. Since each of these vital ingredients can be weighted in forming the overall belief system, is there a minimum weighting which must be assigned to each in order to qualify the person or activity as Software Engineering-related? Will 5% do, or 1%? At what point does the dilution reach the level at which it has so little effect that the activity is no longer Software Engineering?

The degree of skill attained by a particular software engineer can perhaps be related to the quality of his or her fitting the elements of his or her DM to the circumstances by weighting the elements during their merging. For instance, whilst freedom from bugs is a desired attribute in a system (see Sommerville (1992)), does the production of software and its distribution before all of the bugs have been fixed, under commercial sales or management pressures, result in the activity not being Software Engineering? The skill of the software engineer can therefore be related to the exercise of *judgement* during CBSD activities. This may have important consequences for the teaching of Software Engineering

Also, is this affected by the fact that different people, who may or may not consider each other 'software engineers', will define the same desideratum very differently? For example, definitions of 'elegance' may differ between practitioners. What effect does this have on the problems noted by Kuhn (1970, p 202) in inter-paradigm communication, and the need for *translation* between paradigms? Does each software engineer hear the word 'elegant' when a design is considered, and interpret it in his or her own terms? What effect may this have on Software Engineering practice?

These questions need to be addressed at some stage in the research programme started here, and must be answered before we can define Software Engineering properly.

9.2.4.2 Computer-based Systems Development Project Failures

The theory expounded in this thesis has not yet been developed to the stage at which specific failures of CBSD projects can be explained. It may be possible in the future to consider the reasons for projects failing in the light of the DMs underlying the work carried out, and therefore to learn more general lessons from these failures.

This could be done by considering the extent to which an inappropriate or inconsistent DM has affected the project adversely. Additionally, the failures might be examined in the context of the Principle of Uniformity of Systems, considering the possibility of invalid inductive assumptions as a contributing factor.

9.2.4.3 The Lack of a General Set of Development Mechanisms

By generalising from the particulars of a CBSD method, it may be possible to derive a theory to underlie future developments. This could be in the form of a DM and a set of heuristics for selecting elements from it and weighting them to form a suitable set for a particular development. Such a long-term vision has been described in the introductory chapter of this thesis.

The existence of a 'reference collection', of influences on CBSD and of how these influences might be operationalised, would also allow method designers to test the effectiveness of different methods in converting these principles into working systems. It would provide a basis for framing and answering questions in an experimental environment. In considering reports of real-world CBSD experience, it would also allow the consideration of the results of CBSD activities in the context of the influences on the developers. This would inform the process of method development.

The advantages which I see for the computer-based systems developer who chooses this approach and/or uses a predefined method reflect greater information in method choice and use. Such a practitioner would also be able to consider the effect of his or her personal ways of working on the results of

development projects. Such a process might form an early stage of a practical CBSD mechanism based on the Kuhn-based model.

An examination of the DM underlying methods would provide better information for those selecting methods on the effects of the selection decision on the systems developed. An understanding of the underlying paradigmatic influences will enable developers to predict the influences of different methods on the project, and therefore to select a method which is a good fit with what is required.

By giving systems developers an understanding of their practice, examination of even a partial DM would begin to answer the question of how what they currently do affects the results they get.

9.2.4.4 The Lack of Support for the Individual or Informal Developer

The investigation and description of the implicit and explicit contents of the DM of CBSD would promote an understanding of the fundamental reasons for, and effects of, design decisions in formalised CBSD methods, techniques and notations. This understanding could be used as a starting point for the derivation of an equivalent set of DM elements underlying the informal mechanisms often employed to meet the different needs of the individual developer.

The publication of the underlying models, values and influences on CBSD would assist individual and informal developers in selecting and applying tools relevant to their situation. This information, which would in many aspects be as applicable to informal as it is to formal development mechanisms, would provide an understanding of how the tools selected affect the products of the development process. Well presented, the information could be seen as an aid in the informal process of selecting the tools to be used by the *ad hoc* developer. I therefore see such information as a support for developers' experience and understanding rather than as a replacement for it, providing a framework within which developers can make more rationally-based choices of CBSD mechanism.

In the longer term, the predictive mechanism described above, which is likely to be researched initially in the more easily generalised area of method-based CBSD, could be used as a starting point for the derivation of an equivalent to meet the different needs of the individual developer. For example, the weighting given to the need to be able to communicate among the members of a development team is

lessened by only having one member in that team. Additionally, the effect of changes in the DM would be better understood, and the chances would be increased of obtaining a method designed for the sole developer which is coherent, and which eliminates inessential aspects of current methods, whilst still meeting the needs of that developer.

Testing the effectiveness of the implementation of method designers' expected DMs in different methods would permit the analysis of the effectiveness of such implementations and the influence of DM-based side-effects on the design process. Modifying CBSD methods on the basis of this information may allow them to be made more relevant to the individual or small team development environment, specifically by altering the implicit and explicit effects of that method on the mind-set of its users to fit the different set of goals and criteria held by individual or small team developers.

9.2.4.5 The Lack of an Underlying Theory

As articulated in this thesis, the Kuhn-based model does not provide a complete theory to address the question of why CBSD works as it does. It is a model based on the observable phenomena. The Algorithmic Model is a conceptualisation rather than a theory of how the human mind works.

However, the Kuhn-based model presented here may be a first step towards answering the criticism of CBSD of a lack of understanding of how its underlying mechanisms operate. An understanding of how each DM element affects the development process, and how they interact when combined, could form the raw material for the derivation of a theory of how the creative processes in CBSD work, by combining this knowledge with an understanding of the psychological mechanisms.

9.2.4.6 The Lack of a Generalised Teachable Mechanism

Finally, the teaching of CBSD tools could be made more general, and therefore longer-lasting by the application of the work started in this thesis.

The first matter to note is a problem with current craft-based CBSD teaching mechanisms, which all too often start their teaching by training students in the tools and techniques of *one* method. As we have now seen, this will significantly

orient their minds towards one way of thinking about CBSD, as the students absorb the explicit and implicit beliefs forming the DM of that method; compare Petre's (1989) work on programming languages.

This method of teaching will colour students' working lives as computer-based systems developers with the thinking behind one method and its tools. There is a danger that a particular method, and aspects of its DM, will become obsolete, resulting in the risk that the students will become redundant if they are unable to perform personal paradigm shifts to newer, 'better' methods as these are published. The craft-based mechanism of teaching also reduces the chances of being able to take advantage of new thinking brought into CBSD by these new students.

It is therefore important that, before any of this material which influences their thought processes is presented, students are informed of these influences, and of their effects. I therefore suggest that, once the DM of CBSD is sufficiently developed, its content and underlying theory be taught at an early stage in the training of computer-based systems developers. This should be done in a manner which causes the students to understand and question the assumptions underlying any developmental mechanisms with which they are presented in the future, either as students or as part of their professional development during their working lives. In this manner, they may be able to absorb new methods more easily once they are at work, and will understand how these affect the products which they are designing. However, such teaching of the Kuhn-based model must include an emphasis on the beneficial effects of the normal science aspects of CBSD practice, *viz.* that there is a time to accept a set of tools and work within their recognised limitations rather than constantly questioning the selection made.

If teachers were to teach the DM rather than specific methods, then a more portable and durable learning process would be created. This approach would allow the teaching of Software Engineering to be improved, by explaining the implicit as well as explicit effects of different CBSD tools. I suggest that this use of the Kuhn-based model of CBSD is a natural extension and use of the theories presented in this thesis.

9.2.5 The Criticisms of Kuhn Revisited

Based on the work and its conclusions presented above, it is now possible to consider how important the criticisms of Kuhn's work, described previously, are to the Kuhn-based model described in this thesis.

These criticisms and my comments on them are:

- *the DM is not useful as a research tool*; in the case of CBSD, its employment has enabled useful results to be obtained;
- *scientific disciplines advance by other than crisis and revolution*; there has undoubtedly been progress in the effectiveness of CBSD processes in coping with large-scale CBSD work since the Garmisch conference in 1968. I suggest that this progress has been associated with a fragmentation of the DM of CBSD, rather than a change in a unifying DM, as a analogy with Kuhn's scientific revolutions would imply. In addition, a process of gradual rather than revolutionary change may sometimes have operated. Further research is needed into the history of CBSD to see whether there have been revolutions in the discipline, and this is included in the future work below;
- *Kuhn is a relativist*; I suggest that a relativist approach to design decisions is typical of design disciplines such as CBSD;
- *it is difficult to determine the difference between the articulation of different paradigms and different articulations of the same paradigm*; this is perhaps more the case in a science than in CBSD, where most techniques, notations and methods can be at least categorised with respect to some of the broad assumptions made in them, such as the computational model underlying a notation. If we can identify the elements of the DM, we may be able to divide methods, techniques and tools into groups, but the process of identification of different elements, as against shades of the same element, may in some cases be a matter of judgement;
- *Kuhn's views lead him to deny the existence of progress in some scientific changes*; the advances in CBSD practice in managing and co-ordinating processes since the Garmisch conference are not in doubt, but have more recent changes in CBSD mechanisms provided real advances in terms of, say, being able to predict at an early stage the usefulness of computer-based systems in the real world? The Kuhn-based model remains neutral on

whether progress has been achieved, and, if so, on the nature of that progress, and is therefore unaffected by this criticism;

- *Kuhn advocates identifying the scientific community first, then deriving the paradigm from observations of that community*; I have been successful in adopting this approach, first setting out the definition of the activities of CBSD and therefore the community of those who carry it out, and then describing some elements of the DM associated with that community by reference to literature and practice;
- *not all scientists agree on the fundamental metaphysical beliefs within which they work even when they are performing normal science*; the personal input into the DM in CBSD practice, as described in the Algorithmic Model, provides a mechanism for individual viewpoints to colour each practitioner's perception of the underlying DM of a situation which is described in the Kuhn-based model of being like that of a Kuhnian normal science activity;
- *not all elements of belief shared between members of a scientific community are necessarily relevant to that community's scientific work, and they may disagree on some of the elements relevant to their work*; in designating aspects of belief explicit or implicit in a textbook, interview transcript, etc. as relevant to the DM of CBSD, I have implicitly used the criterion of the authors' and interviewees' perceptions of what is relevant to CBSD, in that the content of the books and interviews explicitly concerned the subject of CBSD, in addition to the application of my own discipline knowledge in deciding what is relevant. As for the disagreement between members of the community, I believe that the disagreements which I have found reflect the divided state of the discipline of CBSD rather than a criticism of Kuhn's work as applied here; and
- *Musgrave dislikes Kuhn's use of the term "puzzle-solving" to describe normal science within the DM, preferring the term "problem-solving"*; if this criticism is valid, then the analogy between a Kuhnian science and with CBSD is supported rather than being weakened. One school of CBSD can be recognised by its view of the CBSD process as 'problem solving' – the use by Dahlbom and Mathiassen (1993) of the '8 Queens' problem as a model of one form of CBSD activity provides support for this viewpoint.

9.3 Future Work

9.3.1 Introduction

I describe here the ways in which the work presented in this thesis might be extended and deepened. The objectives and goals of the work are first described in terms of a vision of how the discipline of CBSD might be changed by such work. This is followed by descriptions of the areas of interest and of specific investigations and experiments which might be used to explore these.

9.3.2 Objectives and Goals

9.3.2.1 Short Term

I envisage the short-term extension of the work presented in this thesis as being the deepening of the existing investigations using similar mechanisms. This should consist of trawling through more books, interviewing more subjects and looking at more reports of failures.

As more of this work is done, I suggest that the contents of the DM listed in Appendix 2 will become better defined, the number of new elements found will diminish, the support for existing elements will grow, and as-yet undiscovered subtleties in these will emerge.

At some point, I believe that the return on additional subjects will become sufficiently small for the results to be considered well enough grounded for experimental work to proceed, and at that stage there will be enough information and a solid enough theory for the experiments described below to be performed with an understanding of the likely outcomes.

9.3.2.2 Medium Term

In the medium term, the objective would be to check the results obtained by informal mechanisms, through performing formalised experiments as described in the Work Programme below.

The intention would be to provide formalised support for the informally-obtained information, and to check any conclusions drawn. In particular, it would be easier to:

- assess the actual effect of elements of the DM on the processes and products of CBSD practice; and
- examine an element, or a subset, of the DM, in isolation by comparing the results of parallel work identical in all respects other than the element(s) under consideration,

by controlled experiments rather than by relying on anecdotal evidence. This in turn would enable the building of a theory for CBSD with some measure of predictive ability for the results of developments based on predetermined DMs.

Practical benefits from this research can be hoped to arise at this point, as the results of experiments into the efficiency of mechanisms in methods in delivering the expectations of their designers are published, to inform those who develop and choose CBSD methods, tools and techniques.

9.3.2.3 Long Term

The long term vision of a discipline informed by the extension of the work described here is that of one which might be called with reason 'computer-based systems *engineering*', since it would be based on mechanisms whose effects could with confidence be predicted. The results of a well-designed series of experiments will illuminate many of the underlying influences on CBSD from its tools and situations, and these results will be available to be used in the design of methods and techniques. At this stage, input from other disciplines such as psychology and sociology might also be employed, as suggested below.

In addition, it would be possible to teach students the underlying belief system and the effects of its elements both alone and in combination, providing an education which would need significantly less updating throughout their careers as computer-based systems engineers.

This work could also be extended in general terms, with the long-term objective of building a research-based equivalent theory of the design disciplines to parallel Kuhn's work in the physical sciences, and a 'working model' of this in the form of a version of the Algorithmic Model in whose predictions confidence might be legitimately placed.

9.3.3 A Work Programme

This section describes some proposed research which might enable an advance to be made towards the goals set out above.

9.3.3.1 Testing the Theory

A task to be accomplished as early as possible in the research programme would be to test the validity of the Kuhn-based model described in this thesis.

An experiment specifically designed to falsify the key assertion of the Kuhn-based model of CBSD, that DMs influence CBSD activities, is to examine or perform design studies carried out using methods with different DMs in otherwise identical design situations. This separation of one aspect of a design situation would be difficult to achieve in real world conditions, and such an experiment might therefore be better attempted under the somewhat artificial conditions of a case study. In this case, however, care would be needed to minimise the problem that the effect of the DM of a method on students new to it might be less than might be expected from developers experienced and steeped in the method, i.e. those who have achieved a personal paradigm shift to the method's DM. Similar care would have to be exercised in identifying the influence of prior CBSD work on experienced developers used as experimental subjects. As part of the analysis of the results of the experiment, it would therefore be necessary to remove from the raw results any variations due to each subject's individual approach or prior domain knowledge before comparing the designs. The necessary information may be obtainable from in-depth interviews with the subjects, possibly as part of a selection procedure for experimental subjects.

It is almost inevitable that the designs which form the outputs of these experimental procedures will differ. The test of the hypothesis of the existence or otherwise of the DM of CBSD will be the answer to the question of whether the differences can be traced to differences in the methods' DMs. I believe that differences due to differing DMs will be identifiable under these circumstances.

Investigations and experiments into specific aspects of the Kuhn-based model include:

- testing the accuracy and generality of the DM elements listed in Appendix 2 by selecting a statistically significant number of CBSD theorists and practitioners, and examining their beliefs both in general terms and with respect to specific DM elements previously identified, and the effects of some elements on others, by:
  - performing face-to-face interviews in a more formalised manner than has been possible for this thesis, using closed questionnaires as a starting point and examining the details of the responses in follow-up questions; and/or
  - sending closed questionnaires to them,

  correlating the results obtained after statistical analysis of the responses. These results would provide a quantitative basis for a description of the elements forming the DM. To enhance the statistical results which could be determined by this work, theorists and practitioners could be divided for this purpose into a number of equivalence classes based on some or all of the criteria used to select the interview subjects previously, so that the effects of these influences could be included in the analysis of results;
- testing the reasonableness of the conceptualisation incorporated in the Algorithmic Model as described in Chapter 7 above, using comparisons between similar systems designed under, say, different organisational circumstances;
- quantitatively testing how well the weighting procedure described in the Algorithmic Model reflects actual practice by having statistically valid groups of practitioners and theorists *weight* different elements in terms of their importance to software quality. This process might also allow the investigators to gain an understanding of how theorists and practitioners characterise 'quality' in a CBSD process or product, but this would be a secondary objective compared with the testing of the underlying theory.

  A specific mechanism to achieve this aim is to give each member of the subject group a list of elements, and ask them to:
  - weight each in importance in each of their separate personal and organisational definitions of quality;
  - give any other elements which they believe are important in determining quality; and

- identify any predefined or informal methods, techniques, process models and so on which they use, so that any correlation with the expected DM of the tools they use could be observed.

It has not been possible to perform such investigations as part of the work presented in this thesis, because the elements perceived as being related to 'quality' have only been determined as a result of the work presented here. Without such a list of quality-related DM elements, it has been impossible to use an investigative technique which is closed with regard to attributes of quality.

9.3.3.2 Extending the Theoretical Base

Work which would broaden the scope of examination of the Philosophy of Science to find further analogies which are of use includes:

- a consideration in depth of CBSD in context of the aspects of Kuhn's theories not employed in this thesis, such as:
  - the influences on the DM of nationality, religion, authority of the teacher and so on; and
  - the influences on those who need to choose between competing DMs;
- a re-examination of the analogy with Kuhn's theories in the context of the work presented here, looking for instance at the relative dearth of symbolic generalisations in the list of elements presented in Appendix 2, and considering whether this and other divergences require a rethinking of Kuhn's work to accord more closely with design disciplines or whether more work such as that of Liang (1994) is required to identify symbolic generalisations; and
- examining in detail the writings of other philosophers of science to add depth to the theoretical basis of CBSD. Examples of such sources include:
  - Feyerabend's (1975, 1993) anarchistic approach to science, which has more relevance to the less constrained model of CBSD employed by informal developers than is currently envisaged by the proponents of most predefined methods, and might therefore be potentially more fruitful for the smaller software development organisations, promoting a flexible approach to CBSD (selection of

notations/process models, informal vs. formalised predefined notations, etc.) similar to that of Avison and Wood-Harper (1993); and

- Lakatos' research programmes (Chalmers, 1992, p 80 *et seq.*), and their possible parallels with method development, and with the work required to support systems life-cycles; the work presented here has resulted in a model with clear parallels with Lakatos' model of a belief system formed of an inviolable core and a protective belt, but these parallels should be explored in detail to determine whether useful insights might be gained.

During this work, my experience in using Kuhn's theories leads me to suggest that the ideas gleaned from the Philosophy of Science should not be treated with such reverence that they are left unchanged despite their application in a new type of domain, i.e. a *creative* design discipline rather than an *explanatory and predictive* science. This might militate in favour of an approach which criticises and modifies the philosophies of science in order to provide a theoretical basis more suited to creative disciplines in general, and to CBSD in particular.

9.3.3.3 Extending the Analytical Work

This is a deepening of the existing analytical work on the information gathered to date. The analysis which I have performed for this thesis has not extracted all of the possible benefits from the investigations, and a review in the light of what has been learned so far might include:

- clarifying the investigative work performed to date by, for instance,
  - looking for further evidence for specific elements on the part of authors whose work was trawled, including a reading of other publications by the authors whose work I have analysed and adding to the existing analysis of these individuals, and checking for consistency between their publications; and
  - examining different definitions of key terms such as 'system', 'information', and 'data';

- considering the current grouping of elements under headings in Appendix 2 in the context of:
  - the higher-level paradigms identified by other workers such as Hirschheim and Klein (1989); and
  - any logical or belief-related connections which can be identified between them, forming structures like that relating to decomposition described in Chapter 6;
- examining the differences between the results obtained for CBSD theory and for its practice, to see whether any generalisable differences can be identified;
- determining to which of the many *roles* present in CBSD each of the elements already found relates; I suggest that most of those elements found so far in this work are related implicitly to systems analysis and design – i.e. to roles related to programming in the large, as is explicitly set out in the scope of the thesis work;
- considering the beliefs found to see which level(s) of CBSD, as described in the Algorithmic Model of CBSD, each arises from or is reflected in – discipline, technique, notation, process model, and/or support tool;
- looking at the DM elements in a pair-wise or set-wise fashion to determine the interactions and correlations between them. Such interactions might include the previously suggested theories concerning the consequences of the waterfall model for innovation in CBSD design work. The results of interactions may be as expected or not, and, as suggested in Chapter 7 above, could conceivably cause conflicts within the minds of computer-based systems developers, resulting in a loss of effectiveness in their work. This line of research might be a source of ideas for experiments and comparative trials, or experience investigations of methods with different DMs; and
- looking for the DM elements identified in Appendix 2 using existing published practice-based work such as field reports and the results of action research projects to show how they affect CBSD practice; this is an extension of the work presented here on the results of work on Multiview.

9.3.3.4 Testing, Extending and Refining the Investigative Mechanisms

Before much work to advance the theory can be attempted, the *mechanisms* which have been advanced and examined in this thesis will also need to be tested and improved where they prove to be insufficient. For example, the problem of interview results apparently simultaneously agreeing and disagreeing with particular elements of the DM noted in the discussion on the results of the interviews undertaken for this thesis should be considered before such interviews are used to identify specific element for further study.

Work which might usefully be performed to exercise and improve the investigative and experimental techniques includes:

- having the identical work repeated without modification by other investigators using the same sources where available. Trawling through the same written sources for the same objectives might allow a comparison of different workers' detailed methods and results. However, it will not be possible to interview the same subjects 'cold' twice, and that part of the work is essentially unrepeatable in exact parallel. This problem of lack of exact repeatability might be addressed either by using a second group of subjects as close in background as possible to those whom I have used and comparing the results, or by interviewing a larger number of subjects, correlating the results statistically and comparing these with those which I have obtained,
- having similar work performed by investigators with different backgrounds; particular importance might be given to any differences in results obtained by those investigators with and without real-world CBSD experience, since these differences of themselves may result in the identification of CBSD DM elements by observing the observers;
- looking at the use of investigative mechanisms selected to minimise the input from interviewees' later experience or subjective views of past events; these could include observation-based techniques or immediate subject commentaries as they work;
- developing sets of experimental conditions or mechanisms specifically for examining experimentally the connections between pairs or larger groups of DM elements; and

- examining other types of source of DM elements which have not been examined here and comparing them with the results which I have obtained; this might for instance include
  - looking at any published rationales by the authors of books which I have used in the trawling work; and
  - contacting the authors directly and asking for comments on the results obtained;

  and comparing the results of this work with those which I have presented in this thesis.

9.3.3.5 Broadening the Work within Computer-based Systems Development

It would also be possible to broaden the work within CBSD, to add evidence for existing DM elements and identify further elements. Mechanisms to achieve this include:

- employing the trawling techniques used for the work presented in this thesis, and/or the other possible mechanisms suggested above, on other textbooks and interview subjects to provide evidence for additional DM elements;
- extending the examination of the related work, described in Chapter 8 above, to consider other research along the same lines;
- producing pictures in detail of currently-perceived 'views' of CBSD, e.g. the characteristics of a Tayloristic viewpoint and how it is reflected in a method's belief system, allowing a more detailed critique of methods than is currently possible;
- determining the sizes of the communities associated with different schools of thought within CBSD; this work may reveal some of the lower-level structure of the discipline and thus of the DM;
- examining the work of identifiable schools of CBSD, such as users of formal methods, in this case particularly to see whether any symbolic generalisations can be found in this sub-community;
- extending the information for each element of the DM through investigation and examination:

- the *source* of its influence on CBSD practice – discipline, method, process model, tool, technique;
- whether its definitions diverge at some point leading to the identification of different schools based on these definitions; and
- the *expected effect* it has on CBSD practice;

and by *investigation and experiment*:

- the *importance* of its influence on CBSD practice;
- the *actual effects* it has on CBSD practice, and how these compare with the expected effect – for example, what are the actual effects on CBSD practice of the waterfall model (see above); and
- the *influence on other elements* of this element.

The order in which individual elements are examined might be based initially on the investigators' existing opinions as to the relative importance of the elements, but as information is gathered some prioritisation based on the actual importance of an element both alone and as an influence on other elements will be possible as the fastest way of feeding information back into the discipline practice;

- examining the history of CBSD:
  - looking for any candidates for equivalents of Kuhnian revolutions; as suggested above, the period following the Garmisch conference in 1968 with its apparent air of crisis may be a good first candidate for this analysis;
  - as rewritten by authors at different times; the introductions of textbooks describing CBSD methods, which commonly see that history as a path of uninterrupted progress towards the method which the author presents, might be a useful source for this evidence to support the Kuhn-based model of CBSD theory as school-based;
- examining the underlying bases of modifications made to methods in practice in the context of the Kuhn-based model, to determine, for instance, whether a limit is placed on these changes by the underlying DM of the method itself, and the extent to which any alleged flexibility in the method is reflected in other aspects of its DM;

- examining other reports of failed CBSD projects in order to determine whether any elements of the DM, or interactions between elements, can be identified which might have contributed to those failures,
- building one or more Soft Systems Methodology (Checkland, 1990) models of generic CBSD activities, and considering the importance of the DM of CBSD as the *Weltanschauung* in those models;
- examining the effects on CBSD practice of the inductive basis of the discipline enshrined in the Principle of Uniformity of Systems; and
- looking at the semantic burdens of the terms used in CBSD, to see whether the names given to some aspects form implicit influences in the DM; consider, for example, the quantitative definition of "friendliness" described in Chapter 6 above.

9.3.3.6 Feeding Back into the Real World

The work presented here has been undertaken with a view to advancing the theory and practice of CBSD. It is therefore necessary to consider how the transfer of the results of experiments and investigations to the CBSD community could be achieved. The work required to do this includes:

- presenting the findings of research into:
  - the Kuhn-based model in general;
  - specific DM elements; and
  - the Algorithmic Model,

  to the CBSD theorists and practitioners to whom they apply i.e. in the formal and informal journals and magazines of CBSD theory and practice, to explain inter-school differences and possibly to act as the basis for reconciling some differences of opinion between them;
- developing new or modified methods, tools and techniques informed by or based on the understanding of CBSD gained by applying Kuhn's theories; and
- packaging the Kuhn-based model and the DM elements in a form suitable for teaching to Software Engineering students as a set of principles for designing, selecting and using CBSD methods, techniques and notations,

and producing a syllabus for this purpose, to be used as either a stand-alone course or as a segment of a general Software Engineering course.

9.4 Conclusion – A Paradigm-Based Discipline?

In adopting a model from the Philosophy of Science, I am not saying that CBSD is a scientific discipline. The analogue in Kuhn's work of the current state of CBSD theory is that of a pre-scientific discipline, with schools of thought dividing the community along a number of axes. We may however ask whether CBSD will ever achieve the status of the analogue of a Kuhnian science, with one DM accepted by all practitioners. We might also ask whether seeking this unifying belief system would be a useful goal for Software Engineering theorists, or an unattainable holy grail, another magic bullet. Such a quest assumes that all problems for which solutions can be provided in software are subject to the same set of universal laws. Is this true, even for the significant aspects of CBSD activities? It is in my view better to assume that, at the current state of development of CBSD, there is only an *analogy* drawn between CBSD and Kuhn's view of a science, rather than the suggestion of an identity between the two.

Kuhn notes that, for much of a scientific discipline's life, the periods of 'normal science', the DM of that science controls its agenda, setting out those research problems which it is legitimate to attack and those which are irrelevant or whose approaches are theoretically flawed. Will the enunciation of a DM for CBSD allow the extension of legitimate areas of CBSD research, as elements of the DM are seen not as immutable but as particular options selected, possibly unconsciously, from sets the other members of which have perhaps not yet been considered? The effects of the definition of the DM of CBSD, in fostering debate about the underlying nature of Software Engineering and in pointing out options as yet unexplored, could be considerable.

Appendix 1: The Interview-based Investigation – Interview Questionnaire

Version 1.0 of 19 January 1994

1.1 Introductory Comments

The questions to be answered during the conversations should include the following. The order and grouping of the questions reflects the structure of the information to be obtained, rather than the order in which the question should be covered.

Note that the timings are approximate guidelines for the conduct of the interview; going a bit faster to allow more time for the last sections would be useful!

1.2 The Questions

1.2.1 Background (24 mins; end at 24 mins)

1.2.1.1 Introductory Information (1 min; end at 1 mins)

The interviewer should give

the *date* and *time* of the interview

1.2.1.2 The Investigation Environment (2 mins; end at 3 mins)

Have I previously discussed my work with you?

If so, could you give a short summary of what you think it is about, so that I can take account what you already know about my work when reviewing the results of this interview

To de-contextualise it; to check whether they may be aware of the implications of the questions

1.2.1.3 The Working Environment (1 mins; end at 4 mins)

In relation to the current or most recent computer development you work in or have worked in

Do/did you work for an organisation, or by/for yourself?

If you work(ed) for an organisation,

Is/was its main function the development or tailoring of computer software? If it is not,

what is/was its main function?

what is/was the relationship of IT to the principal business of the organisation?

Is/was the organisation profit-making or non-profit (education, etc)? *(may give some indication of the types of external constraints on the developers)*

1.2.1.4 The Participant (20 mins; end at 24 mins)

1.2.1.4.1 Background (5 mins; end at 9 mins)

Before your career in computing began:

general education

pre-computing career path (if applicable)

Career in computing:

how did you get into computing?

development of career in software

what was your first job in computing?

development of career since then

education

theoretical

formal education (school, post-school)

practical - formal

courses attended

courses taught

When did you last attend a course in any aspect of your work?

What was the course about: management, analysis, design, implementation, etc.?

How long ago was it?

practical - informal (craft)

learning by doing *(note the issue of manuals + text-books vs. trial-and-error learning)*

learning from craft masters (such as colleagues)

If you are, or were at some point in your computing career, a programmer (= coder) *(this may be confirming facts already elicited)*

was this initially as part of your job, or as a hobby?

was this before you became a software designer? If programming was initially a hobby, did this influence a later move into computing as a career?

what was the first programming language you learned or were taught?

what is currently your favourite programming language?

1.2.1.4.2 Current Position (5 mins; end at 14 mins)

Are you currently working in computing?

If not, what is your current career? *Also, the interview should relate to the last mainly computing-related position held.*

How long have you been/were you in computer systems development?

How much of that time was spent developing the types of systems you are developing now?

How much of the system life cycle are you personally responsible for?

analysis
design
implementation
installation
maintenance/enhancement

How much of the work do you do yourself in each phase?

When did you last write a program as part of a system development with which you were associated?

How big was the program?

Did you write it because you had to *(perhaps as part of your job, or to meet a deadline)*, because you wanted to, or for another reason?

1.2.1.4.3 Attitude-forming Information (10 mins; end at 24 mins)

1.2.1.4.3.1 Defining Software Engineering (5 mins; end at 19 mins)

How would you define

the term 'Software Engineering'?

the discipline of 'Software Engineering'?

What is your attitude to 'Software Engineering' as you define it?

1.2.1.4.3.2 Defining Methodologies, Techniques etc. (5 mins; end at 24 mins)

How would you define a systems development 'methodology'? (or 'method' – what term does the subject prefer? what does he/she use?)

What was the first methodology you used or were taught?

What is your current favourite methodology?

What were the first modelling technique and notation you were taught? (at which level of modelling is this answered? also note cross-reference to *Techniques and Notations* below – currently 3.4.2)

What is your current favourite modelling technique or notation?

What do you view as the relationship between these techniques and notations, and methodologies?

Now present a short explanation of the differences in my view between 'methodologies' and 'techniques' and between 'formal' and 'informal'. Then see if any of the questions need to be asked again or any answers clarified

1.2.2 The Development Environment (5 mins; end at 29 mins)

Again, this relates to the subject's current employment, or the most recent relevant employment

What types of computer systems are/were you involved in developing?

How large are the systems you are/were involved in developing?

Do/did you develop systems by yourself, or as part of a team?

If the development is/was a team effort

where do/did you fit into the organisational structure?

how do/did you communicate with other team members? *verbally, informal documents, formal documents*

how do/did you co-ordinate your work with other team members?

1.2.3 Managing the Software Development Process (10 mins; end at 39 mins)

Note cross-reference to Methodologies and to Computerised Tool Support for Software Development below

What tools (intellectual, computer, etc.) are used to

estimate the resources required before development work starts? *('none' is a valid answer, as is 'back of envelope'!)*

monitor progress and resource use against the budget?

How do you as systems developer *(or single developer in 'developer' role)* measure progress during a software development?

How do you as manager, or how do your managers, *(or single developer in 'manager' role)* measure this progress?

Do your development projects 'succeed'? How do you define 'success' in this context? *(Enumeration of examples is permissible!. If examples of successes are give, probe for examples of failures)*

If your projects do succeed,

to what extent do you think that is this due to good management or good development techniques?

to what extent do you think that this is despite bad management techniques?

(how important do the front-line troops view management to be?)

If they do not succeed, why don't they?

1.2.4 Systems Development Tools (50 mins; end at 89 mins)

These question should be asked both in general terms, and in the context of an actual recent system development. The answers should reflect both the theoretical and the practical situations, which may differ.

Apologise for any repetition of earlier information seeking! Also may need to reprise distinction between 'methodology' and 'technique/notation', and between 'formal' and 'informal'

1.2.4.1 Methodologies (10 mins; end at 49 mins)

Note cross-reference to Managing the Software Development Process above (3.3)

Do you use a formal (predefined) methodology for your systems development work

a process model; an ordering of specific tasks performed or repeated in a certain order to develop or maintain software?

one or more techniques for analysis, design and/or implementation?

If so

is the methodology an external standard or an in-house development?

if it is an in-house development, how was it developed? *(modification of existing methodology, or new design?)*

process model

model selected

criteria for choosing it

choice of techniques

techniques selected

criteria for choosing them

If you do not use a pre-defined methodology

why not? *(ignorance, conscious decision (but why?)*

1.2.4.2 Techniques and Notations (15 mins; end at 64 mins)

1.2.4.2.1 Formal (7.5 mins; end at 56.5 mins)

Do you use any predefined system development techniques and/or any formalised modelling techniques during the analysis and/or design process?

For these, if they are pre-defined by the methodology, and if they are used as defined by the methodology then just check that they seem to be correctly used [if I can tell!]. Try to avoid too much detail

If so

what do you use?

do you use them consistently, or only when informal techniques fail?

For each technique

describe it *xref to descriptions of informal techniques*

how did you find out about it?

is it an external standard or an in-house development?

if it is an in-house development, how was it developed?

do you use it consistently, or only when informal techniques fail?

why do you use it

at all?

as against any other technique?

preference, efficiency, management diktat?

1.2.4.2.2 Informal (7.5 mins; end at 64 mins)

Do you use any informal or *ad hoc* notations and/or modelling techniques during the analysis and/or design process?

If so

what do you use? *(may need to explain by example)*

do you use them consistently, or only when formal techniques fail?

For each technique

describe it *xref to descriptions of informal techniques*

how did you find out about it?

is it an external invention or an in-house development?

if it is an in-house development, how was it developed?

do you use it consistently, or only when formal techniques fail?

why do you use it

at all?

as against any other technique?

preference, efficiency, management diktat?

1.2.4.3 Implementation (10 mins; end at 74 mins)

Which computer language(s) if any do you personally use for implementing systems?

What programming languages do you choose to use when you have the choice? *[trying to determine the current favourite language, to show whether they have any advantage from changing to more recent methods, e.g. David Samuels, who uses back-of-envelope design but uses C rather than C++ so would have less assistance from an object-oriented modelling notation]*

Which computer language(s) does your organisation use for implementing systems?

Does your organisation have standards for the

choice of implementation languages?

if so, what are the criteria for selection?

the ways and styles for the use of these languages?

if so, what are the criteria for selection?

If so,

do they apply the languages and the styles

at all?

consistently?

could you give examples of these standards in action?

what is your opinion of these standards?

1.2.4.4 Quality in Software and its Development (15 mins; end at 79 mins)

This is the key part of the questioning, but it should be 'lost' among the rest of the questions. Bring the subject of quality up casually, as if it is merely one part of the investigation, not the core, and see what emerges. Look for differences between personal and organisational descriptions, bearing in mind that the 'organisational' views as given are filtered through the individual's mind-set. Also note the need to separate process and product quality, though this should be done later and not emphasised as part of the interview

All the 'questions' listed below are guidelines; they show what is to be found out, not what is to be asked. Try not to ask them directly, but seek to have the subject give the answers to them in the discussion. Need to start the ball rolling with something like:

An emphasis is being placed nowadays on the notion of 'quality'. What does this mean to you?

How do you personally define 'quality' in a software product?

How would you distinguish a 'good' (high quality) design from a 'bad' (low quality) design?

How do you personally define 'quality' in a software development or maintenance process?

How do you achieve quality in your own work?

How do you check that quality has been achieved?

Do any aspects of your working environment (particularly the tools you use) militate

in favour of the achievement of quality?

against its achievement?

How does your organisation define 'quality' in

software products, and

the process of developing them?

Do you agree with this definition?

If not, how significant are these disagreements and in which areas do they arise?

How does your organisation go about achieving 'quality' in its software products and processes?

Do you believe that these mechanisms are suitable and achieve their objective?

If not,

how significant are your disagreements with your organisation?

in which areas do they arise?

how would you change the development mechanism to improve the quality of the product?

How does your organisation check that quality has been achieved in a software product?

1.2.4.5 Computerised Tool Support for Software Development (10 mins; end at 89 mins)

Do you have access to any advanced (i.e. beyond office standard tools such as word processors, pens, pencils, paper, plus the occasional template for hard drawing) systems development tools? *Note xref to 'Managing the Software Development Process' above*

If so,

what are they?

what do they do; what parts of your work do they help you with?

do they hinder any parts of your work? if so, which parts? *(examples?)*

do you use them?

If so

why do you use them? *preference, efficiency, management diktat?*

If not

why not?

Are you satisfied with the performance of your system development tools?

If so

what would make you change to a different or enhanced tool set?

If not

what is wrong with them?

what would improve them?

1.2.5 Any Other Business

Are there any other matters which you would like to raise in the context of this interview?

If so, what are they?

Appendix 2: Elements of the Disciplinary Matrix

2.1 Introduction

I set out here a list of the elements of the DM of CBSD which have been proposed and/or found to date from all sources. This list is not intended to be 'the' DM of CBSD. I have no opinion as to whether all of the elements forming the DM will ever be identified and documented. However, it is in the nature of the DM that its elements have their effect on the thinking of practitioners even if (in some cases *especially* if) they have not been recognised and made explicit.

Most of these elements are supported by evidence from sources already described in this thesis. Some elements are derived from the investigative work described above; others are supported by evidence obtained as a by-product of the investigative mechanisms, such as those which were noted whilst not actively being sought during the second phase of the trawl through textbooks.

The headings of the columns in the table have the following meanings. The *'Element'* column contains a description of the DM element under consideration. The other columns describe the source(s) from which the element has been identified. These are:

- *Bk1*: book trawl, phase 1 – the number of sources in which the element was identified;
- *Bk2*: book trawl, phase 2 – the number of sources in which the element was identified as being agreed with: for elements common to all CBSD workers, this is also the number in the element was found; for dividing elements, both agreement and disagreement are noted, as reflecting evidence supporting the identification of that element; in both cases, sources for which the opinion was unclear ("?" in the table in Chapter 5) were ignored in this table;
- *Intvw*: the number of interviews with CBSD practitioners in which the element was identified;
- *Act'n*: identified during the trawl through a report of action research into the Multiview framework (Avison and Wood-Harper 1993);
- *Fail*: identified during the trawl through the report into the failure of the London Ambulance Service system (LAS 1993); and

- *Other*: identified from other sources, possibly anecdotal, or derived from informal examinations of the sources used for trawling or from other literature.

For each element, the numbers of interviewees and books is given in the table; where an indication such as '2.3' is given, this represents strong support from two sources and weak support from three. Cases in which dividing elements are disagreed with by sources are indicated by negative numbers, similarly split into weak and strong negative evidence where applicable; such an occurrence might be indicated by '−1.2'. A case in which both positive and negative evidence has been found is indicated by '2.3−1.2'.

As stated in the main text of this thesis, the appearance of a number in the 'interview' column does not imply that the interviewee believes in this him- or herself. It is intended rather to demonstrate that such a belief or model is prevalent among some or all of the CBSD community. This might be from the interviewee him- or herself, the methods or tools he or she has used, his or her employers or clients, or his or her view of the discipline of CBSD in general.

In cases where no evidence from the investigations or other sources is indicated, the existence of an element has been suggested by my experience, introspection and/or speculation. This in turn has been based on my own knowledge and experience of CBSD theory and practice, a consideration in that context of the elements of the DM as they have been identified, and/or the explicit process of analysis described in Chapter 6 above in the context of beliefs related to the decomposition of models of problems and solutions into sub-models.

Finally, it will be noted that some of the elements listed earlier in this thesis as results of the book trawl have had their wording modified or reworded as a result of conclusions drawn from the subsequent investigations also reported in this thesis, or from further analysis of the current state of CBSD resulting for instance in the splitting of an element originally found in the textbooks into a number of lower-level beliefs.

Due to the volume of data collected and comments written during my investigations, only the elements have been listed here, rather than the full

definitions, discussions and rationales which lie behind many of them. Three elements are described and considered in more detail in Chapter 6 of this thesis.

2.2 Elements Common To All Computer-based Systems Developers

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
2.2.1 Symbolic Generalisations						
• Rules from Complexity and Computability Theory						
The Church-Turing Thesis.						Science base
Computability: the problems dealt with by computer-based systems developers are computable; there is a theoretical solution to the problem.						Science base
Complexity (big-O notation): the problems dealt with by computer-based systems developers are tractable.						Science base
2.2.2 Metaphysical and Heuristic Beliefs and Models						
• General Metaphysical Beliefs						
A compromise must be made between conflicting priorities during the systems development process.	2	4	3		1	Simon (1975)
The Principle of Uniformity of Systems.			3.1	1		Logical analysis, Robb (1994)
Godel's Incompleteness Theorem.						Science base
Abstraction as a mechanism for design is a good thing.						
Abstraction as a feature of a design is a good thing.	2	3				
Abstraction as a feature of an implementation mechanism is a good thing.						
There is an optimum level of abstraction in a design.			2			
Things should generally be easier and/or cheaper to rework than to replace.[1]						

[1] I am grateful to Celia Gould for this insight.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Beliefs about How Software Development is 'Done Properly'						
There is a recognisable difference between hacking and doing it properly.			2.1			
The difference between hacking and doing it properly results in different products.			1			
Doing it properly rather than hacking results in better products.						Introspection based on investigation results
• Beliefs About the Nature of the Real World and How it Can be Modelled						
The analytic basis of information systems.						Gregory (1994)
The world makes sense.						Robb (1994), but see Gregory (1994)
The need for Bounded Rationality.						Simon (1975)
The scale of a problem affects the best ways of addressing and/or solving it.			7			
All members of a class of data can be regarded as functionally identical.						Beeson (1994)
All problems of a particular type can be regarded as functionally identical.			1.1			
The world can be changed by the effects of a computer-based system.			1			
• Beliefs Concerning The Internal Structures of Computer-based Systems						
Computer-based systems should be built up from components rather than being monolithic single programs.	2	4	3			
Decomposition and composition do not alter in a negative fashion the nature of the thing broken down or built up to a significant extent (or the benefits of such decomposition or composition outweigh the disadvantages).	2	4				
• Explicit Beliefs About, and Implicit Influences on, the Software Process						
The Petre Effect.			0.1			Petre (1989)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Busy Beavers, and the need to satisfice.						Simon (1975)
There is a practical (as against theoretical) 'solution' to the 'problem'.			1			
The problem must be understood before a solution can be devised.			2			
It is possible to obtain the information required to design the appropriate system for the circumstances.						Robb (1994)
Developers and users can be brought to apparent agreement on ends and means.				1		Robb (1994)
Users can be brought to apparent agreement amongst themselves on ends and means.			2	1		Maguire (1994, p 4)
We're going to have to look after what we've developed.			1			
• Beliefs about Method-based Computer-based Systems Development						
Change in the existing situation is inevitable as soon as a method is applied.						
The benefits of a change suggested as an outcome of an application of a CBSD method outweigh the costs.						Maguire (1994)
There is an optimal level of detail in a predefined CBSD method.			1			
• Beliefs Concerning Faults in Computer-based Systems						
Testing is a good mechanism for detecting faults in computer-based systems.			1		1	
Testing needs to provide positive rather than negative assurance of correct system operation.					1[2]	
It is easier to test small things for faults than big things.			0.1			

2 "The Chief Executive's report ... states that 'There is no evidence to suggest that the full system software, when commissioned, will not prove reliable.' In mission critical systems negative assurance is not enough. There should always be positive assurance through QA and testing that the system will perform to specification." (LAS, 1993, p 35, para. 3102)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Computer-based systems should be designed to be easily debugged.						
The requirements of computer-based systems should be correctly specified.			1			
The specifications of computer-based systems should be correct with respect to the requirements.			1			
Computer-based systems should have as few faults (bugs, wrong problems addressed etc.) as possible given other constraints.[3]			4			
Computer-based systems should be designed so as to minimise the effects of faults (bugs, wrong problems addressed etc.).	3	3				

- Beliefs about the Future Course (Maintainability) of Computer-based Systems

The world is not made static by the introduction of a computer-based system: it will continue to change.			4			
Computer-based systems should be designed to be maintainable.	3	4	6			
Computer-based systems should be designed in such a way as to be changed as easily as possible (design for modifiability).	1	4	4			
The purpose (rationale for the continued existence) of the system will not change sufficiently as to invalidate the reason for the system before the cost of implementing it has been recovered.			1			Robb (1994)
The internal aspects of computer-based systems should be designed in such a way as to be understandable by other than the original designer.			3			
The continued usefulness of any computer-based system should not be reliant on the knowledge of a limited number of people.			2			cf. U.S. Army Payroll Syndrome

3 Compare the later dividing element on computer-based systems having NO faults.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
There is an optimum level of detail in systems documentation.			1			
There is an optimum level of detail in user documentation.			1			
• Beliefs Related to Portability						
Computer-based systems should be designed to be portable.	2	3				cf. Naur and Randell (1969, p 56, p 118)
Systems should be designed to be flexible in use.			2			
• Beliefs about 'Quality'						
Computer-based systems are not intrinsically bad.						Booch (1994)
A CBSD process or product has a level of 'quality' which can be compared with that of other processes or products.						Argument in Chapter 6
The internal structure of a software system is a significant feature in determining its quality.			4			Jackson (1983, p 353)
The external aspects of a software system are a significant feature in determining its quality.			1			
Changes to a computer-based system should not compromise its structure.			0.1			
Software should (at least) meet its specification.			4			
• Notations and their Underlying Models						
All software design is with respect to a particular, identifiable computational model.	4	4	3			
A notation (as used in practice) is an expression of an underlying computational model, which has an independent existence from it.			3			
A notation (as used in practice) is an expression of an underlying virtual machine, which has an independent existence from it.			3			

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
The essence of a computer-based system can be captured by the application of a small number of computational models in the form of graphical or verbal notations.			2			
A distinction can be drawn between the set of elements forming a notation and the style in which that notation is used.			1			
Notations should be designed to be readable by humans.						
Notations should be used in a manner which makes them easily readable by humans.			1			

- Beliefs about The Nature of the Technical Base

Stability is a good thing in the right place.						
Change in technical matters in the right place is inevitable and good.						
Quantitative changes can result in qualitative changes.						

- Beliefs about Standards

Standards are a good thing.			3			
Widely-held standards are a good thing.			1			

2.2.3 Values

Honesty.			4			
Correctness of system outputs.			3[4]			
Consistency of System Design:						
. across platforms.			1			

---

4 I take this element to be common to CBSD theory and practice. Note, however, subject F's view (transcript, p 16): "what's actually *shown* on the output reports erm is of secondary interest. 'By god, that looks smart' they say. 'I paid a lot of money for that and it's really well worth it. Doesn't it look good?' erm ... eventually of course the results have to be right". The subject's last comment seemed to be an acceptance of the 'party line' of common CBSD belief. Despite the initial negative tone, I therefore regard the entire comment as recognising that computer-based systems should operate correctly, and have included it in the figure given.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
. within an application environment.			2			
. across systems within an organisation.			5			
. within and across development environments.			1			
. within a system.			5			
Data integrity.			2			
Creativity.			4			
Clarity.			1			
Aesthetics of internal construction.			1			
Novelty:						
. novelty in the problem is important in making CBSD interesting.			1			
. novelty in the solution is desirable in CBSD.			1			
Firmitas: reliability, structural integrity, ...			4			Vitruvius (1931)
Utilitas: systems should be useful.						Introspection, Vitruvius (1931)
Usability: systems should be easy to use.			4			
Relevance: systems should be relevant to the situation.			1			Checkland (1990)
Minimisation:						
. in theory: faster is better; smaller is better; cheaper is better.						Economic imperative
. as operationalised:						
- elegance:						
in model building.			1			
in systems design.	2	2.1	1			
in building systems (the process, not the product).			1			
- efficiency:						
the system as a *minimalist construct*.			1			
the imperative of the economic system.						

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
the system should be designed to run as fast as possible given any other relevant constraints.	1	1	2			
- simplicity in documentation.			1			

2.2.4 Exemplars

Commonly accepted exemplars of software products.						Personal domain knowledge
The analyse-synthesise-evaluate microcycle.						Personal domain knowledge
OS/360 - the first large-scale computer-based development.						Brooks (1975)
The 80/20 Rule and its relatives.			1			

2.3 Elements Dividing Computer-based Systems Developers Into Schools

2.3.1 Symbolic Generalisations

Rules for the calculation of predicted cost budgets and timescales:						
. numeric tools for predicting costs.						COCOMO: Boehm (1981)
. numeric tools for predicting timescales.			0 - 1			

2.3.2 Metaphysical and Heuristic Beliefs and Models

- Beliefs About the Fundamental Nature of Software Engineering

Software Engineering is like other engineering disciplines.	1	2	- 2			
'Hard' vs. 'Soft' approaches: there is a balance between hard science and socio-technical viewpoints in CBSD development styles.			1			Downs *et al.* (1992), Makepeace (1993)
Software Engineering is only possible using predefined mechanisms.			2			

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Software Engineering is only possible with strong management controls.			1			
• Beliefs About the Current State of Computer-based Systems Development						
Compromises made between conflicting priorities during the systems development process do not necessarily affect product and/or process quality.			1			
The usefulness of a CBSD method can only be measured by its success in developing systems i.e. pragmatically.			1			
• Beliefs about the Place of the Developer						
What should the 'developer' do - solution builder, users' 'partner'?, IT facilitator and/or playing the user.			1			
The people who do the analysis and/or design are technically minded and/or technically competent.			1 - 1[5]			
When all else fails, read the manual.						Introspection, observation, personal experience
Software development should be fun.			2			
Developing new systems is more fun than maintaining existing ones.			1			
• Beliefs About the Software Process						
A formalised software development process is always a better software development process.			1 - 1			
Using a formalised, predefined software development process costs more than an informal mechanism.			2			
Informal development techniques are sometimes more useful than formalised ones.			1			

5 Both from the same interview subject (Subject E), in different parts of description of his working life history.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Truth-seeking vs. (overt?) satisficing.[6]			1	1		Simon (1975), Dasgupta (1991)
Given two of input, process, output, the other one can be defined sufficiently well to build a system model.						Jackson (1983)
The 'needs' and the 'wants' of the user are different, and can be differentiated.			1			
Computer-based systems development:						
. is like 'research'.			2			Connaughton and Dampney (1994)
. is like 'product development'.						Connaughton and Dampney (1994)
A 'specification' can be described stating what a software entity should do and/or its structure before it is implemented.			- 1			
The phases into which the development of software can be divided should be differentiated in some manner.						SSADM (Downs *et al.*, 1992)
A 'process model' can be identified in any software development method which is separate from the other specifics of the development method.			6	1		
Project management and development techniques are of comparable importance in promoting project success.			1.1			
The tools used for software development (whether computer-based or not) need to be flexible.			3			
Computer-based tool support for software development is essential.	- 0.1	3.1				
The advantages of computerised tool support for software development outweigh the disadvantages.			3			

6 The apparent duplication with the common element noted previously of 'Busy Beavers, and the need to satisfice' is in fact a different between my belief that satisficing is necessary, as against the belief among some computer-based systems developers that there is an ultimate 'truth' value for the results of a systems development and that this truth can be found, for example by applying formal methods.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
You can take some part of the software life cycle and deal with it ignoring the rest of the life cycle.						
Software Engineering styles should follow the latest fashion.	1	3	0 - 1	0.1		
The goals and objectives of (a stage of) a computer-based system's development can be clearly and unambiguously defined in advance of work commencing.			1.1 - 1			
The goals and objectives of (a stage of) a computer-based system's development can be clearly and unambiguously defined at all.			1 - 1			
Users know what they are doing.			- 1			Own domain knowledge
Users know what they want when they see it.			1			
Users know what they want before they see it.			1 - 2			
The users want the system being developed.				1		Assumed in user-centred design
The users will need to support the process of developing the system.			1	1 - 1[7]		
The support of users assists the process of developing a system.				1		
Initial prototypes should be the basis of final software implementations.			1			
Software development and software maintenance are the same thing (or the same people should develop and support a software product).			- 1			
Design styles:						
. from user interface in.			1			
. from system internals out.						
Checking of work done in a computer-based systems development is important.			1		1	QA approach

7 The apparent contradiction raises from the results of experience documented by Avison and Wood-Harper (1993, p 267); an un-cooperative user "... contradicts the arguments of 'pure' Multiview ... in which it is assumed that it is always possible to use responsible participation in information systems development."

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Computer-based systems should be made live as a whole (i.e. disagreement with incremental delivery/implementation).					1 - 1[8]	

- Beliefs About Group and Individual Development, and Staffing Computer-based Systems Development Programmes

The task-based model of CBSD: analysis, design, implementation, ...			1 - 1			
Conceptual skills are more highly valued than operational skills.			1			Jackson (1983, p 338)
The best analysts and/or designers are former programmers.						
'Software Engineering' can only be carried out by groups of people.			1 - 1			Sommerville (1992, p 2)
The group worker and the individual worker have different needs from a CBSD method.			- 0.1	- 1		
Ceteris paribus, the results of team and individual developments are not necessarily the same.				1[9]		
The work of software development theorists is irrelevant to the 'real world'.						Anecdotal from talking to developers
The work of software development theorists is irrelevant to individual developers.						Anecdotal from talking to developers
Group working is important, or group working is more important than individual working, or even supporting group working is more important than supporting individual working.			1			

---

8 The one-phase implementation of the original system was criticised by the inquiry team (LAS, 1993, para. 1007(l)), incremental delivery was recommended as a way of delivering the replacement system (LAS, 1993, paras. 1009(c)(vi) and 5004(f)). The three phase programme recommended is detailed in LAS (1993, paras 5018-5041).

9 "We observed that the group results were usually more constrained and conservative, where the individual work was often more creative and flexible. A trend worth noting was that many of the students found that, as they used the methodology more, they began to use it more as a series of techniques rather than as a framework for creative thought. As a result, their products began to show less creativity." (Avison and Wood-Harper, 1993, p 150)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Successful group working is contingent on formalised communication between group members.			3			
Successful group working is contingent on communication between group members, whether formalised or informal.			4 - 1[10]			
Getting the requirements right is more difficult than design.			1			
Getting the design right is more difficult than implementing it.						
The people who check a piece of work should be independent of those who did it.			3			QA theory
Work on computer-based systems should be staffed internally rather than externally.			1			

- Beliefs Related to Methods for Computer-based Systems Development

Differing definitions of 'Method'.[11]						
Methodical working is important in CBSD.			1			
A CBSD method requires sanction from, or is given added authority by, authoritative authors.			1.1			
Adopting a CBSD method is a good thing for marketing reasons.			2			
The adoption of a predefined CBSD method is a good thing.			1			
A method can be devised which is suitable for the development of more than one system.			- 1			
A method can be devised which is suitable for the development of all types of system.				- 0.1		Extension of investigation results
The size of the problem affects the method to be used.			2			

10 Subject H appears twice here; providing evidence for his own belief and for his employers' disbelief

11 Further investigation is required to establish the different values of this element in general currency.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Computer-based systems can be developed using 'common sense'[12] mechanisms.			1			
Some methods are especially suitable for certain types of systems.						Extension of observation
Some methods are especially suitable for certain types of environments, *or* the method should be selected on the basis of its suitability for the environment.						Example from Maguire (1994)
Some methods are usable only for bespoke systems development (as against package products).						Extension of investigation results
A software development method has to fit the situation, *and/or* the choice of method is dependant on the situation.			1			
A software development method has to fit the organisational context, *and/or* the choice of method is dependant on the organisational context.			2	1		
A set of techniques, notations, development tools, process model and management tools can be selected in advance of a software development, and these tools will be sufficient to perform that development.			3.2 - 1			Benyon and Skidmore (1987)
A complete software development method (including any or all of techniques, notations, development tools, process model and management tools) can be rigidly defined in advance of a development project.			- 1.2	1		
A continuum: (various values for) the level of detail at which a method should describe its component activities and models.			1			
All of the tasks and models required by a predefined method should (or must?) be completed.			- 3	- 1		QA approach
A CBSD method, tool or technique needs to be rigidly defined before its products can be checked by a quality assurance function.			1.1			

12 Direct quote from interview subject (subject G transcript, p 4)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
The benefits of flexibility in tool or notation selection and/or substitution in a methodology outweigh the disadvantages.			2 - 1	1		Benyon and Skidmore (1987)
The tools employed define or constrain the design of the process model of a method to some extent.				1		
There is some way in which the techniques, notations, development tools, process model and management tools need to 'fit together' to form a successful method.			2			
Different systems development tools, techniques and/or notations produce sufficiently similar results for substitution of one for another to be permissible.			2	1	1	'Toolbox' mentality (Benyon and Skidmore 1987)
Some modelling notations are better for analysis than others; some are better for design than analysis.						Informal examination of existing methods
Module specifications are to be implemented unaltered by programmers.			2.1			
Programmers do not need to see the big picture in order to do their work.			1			
Programmers are (or should be expected or allowed to be) less creative than analysis and designers.			2.1[13]			
Tools for software development should be easy to use.			1			But see hair-shirt attitude of UNIX developers
I trust the tools I use to develop software.			1		1[14]	cf. Booch (1994)

13 Subject F both supporting and disagreeing; there is some self-contradiction in what he says, but the transcript shows that he has recognised and thought about this issue.

14 In the context of the use of buggy versions of implementation software to develop a critical system: "It is ... probable that the unproven combination of Visual Basic within Windows 3.0 led to some of the early system failures. Our own simulation of certain Visual Basic routines within Windows 3.0 has resulted in some unexplained system crashes. Running the same routines under Windows 3.1 resulted in no such problems." (LAS, 1993, para. 3128).

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Beliefs Related to the Assumed Priority of One Method over Others						
My CBSD method (the one I designed or the one I know) can replace any other development mechanism.						
My CBSD method (the one I designed or the one I know) is the best there is.			1			
The method I use is better than my own judgement.			1 - 1			
• General Beliefs about Notations						
The essence of a computer-based system can be captured using graphical and/or verbal notations.			1			
The complete substance of an analysis or a design can be captured in one notation (or there is one computational model which can capture all aspects ...).			- 2			
Notations should be completely defined in advance of their use.			- 1			
Notations can be evaluated in abstract terms.			1			
Notations should be unambiguous when read by humans.			1 - 1			
The meaning given to each element of a predefined notation by its users is valid in the context of the meaning given to it by its authors and field testers (and QA checkers!).						
The meaning given to data stored in a model using a predefined notation when accessing it is valid in the context of the meaning given to it by its authors (and QA checkers!).						
Influences on the design of graphical notations:						
. There are 'good' values for the minimum and maximum numbers of elements in a diagram.[15]	1	0.1				Miller (1956)
Self-modifying systems are specifically considered to be bad.	1	0				

15 Or is this an unavoidable, common element drawn from a psychology 'science' base (Miller, 1956)?

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Beliefs about the 'System'						
Concerning different definitions of 'system'.[16]						
The 'system' can be well defined before implementation is completed.						Maguire (1994)
The boundaries of 'the system' are well-defined, and can be identified with sufficient accuracy to allow a computer-based model of the system to be developed and used.						Robb (1994)
• Beliefs Concerning the Nature of Information in the System						
Different definitions of 'information'.[17]						Beeson (1994)
Different definitions of 'data'.[18]						Beeson (1994)
Where should 'meaning' be given to 'data'?						Beeson (1994)
• Beliefs Concerning Documentation						
Computer-based systems should be well documented.				3 - 1[19]		
Interactive computer-based systems should be self-documenting on-screen.	1		2	1		
• Beliefs Concerning Portability						
Computer-based systems should be designed to be portable within predefined environments.				3		
The (possibly proprietary) implementation platform which I choose for my application will be available for the lifetime of that application.				2		
• Beliefs Concerning the Nature of the User Environment and the Place of the User						

16 Future work will be needed to specify these; see Chapter 9.
17 Future work will be needed to specify these; see Chapter 9.
18 Future work will be needed to specify these; see Chapter 9.
19 The disagreement comes from a worker in an academic research environment.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Who is 'the user'?						
. the people who hit the keys.						
. the people who read the reports generated by the system.						
. the managers of the people who hit the keys.						
. the people who pay the bills in a bespoke development.						
. 'the market' for a package development.						
The 'user' is important in the development cycle.			2		1	Maguire (1994); Boland (1987)
The relative importance of the end user and end user management.						
The system's organisational environment is a hierarchical mechanistic bureaucracy.						
Attitudes to the end users of a system:						
. master.						
. friend.						
. moron.			1			
Users can understand the technical languages in which models built during CBSD are described.			3	1		
Users who understand the technical languages in which the models built during CBSD are described will not interfere with the technical issues.			1 - 1[20]			
Communication with the end users of a system is important.			1.1			
The degree to which users should be involved in the development process: a continuum.						Socio-technical approaches
Models of interaction between developer and user:						
. negotiation.			1			
. user as developer.						
. developer as dictator.						

20 Both from same subject; using a method whose designers acted as though they believed in this element, he provided a counter-example (Subject H, transcript, p 29)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Attitudes to Software Development						
Who owns the problem?						Jackson[21]
. Developers.						
. Development managers.						
. End users.						
. End user managers.						
Who owns the solution?						Hirschheim and Klein (1989)
. Developers.						
. Development managers.						
. End users.					1	
. End user managers.					1	
Who owns the process:		1				
. Developers.						
. Development managers.						
. End users.		1				
. End user managers.		1				
Orientation of development mechanisms towards:						
. product.						Blum (1993)
. problem.						Blum (1993), Bansler and Bodker (1993)
. solution.						
The specification is there to be exceeded, not just met.		2				Schach (1993, p 453 *et seq.*)
Maintenance of existing systems is more difficult than developing new systems.		- 1				
Software development and maintenance mechanisms should make changes difficult for the developers.		0.1				
Software development and maintenance mechanisms should make changes appear to the users to be difficult.		0.1				

21 My notes of a lecture given to the British Computer Society North London branch by M.A. Jackson, 1994

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Beliefs Related to Faults in Software						
Faults will inevitably occur in software and should be allowed for: fault tolerance.			1			Jackson[22]
Complete freedom from faults is a valid objective of CBSD.			- 0.1			Formal methods
Faults can be designed out of a system.						Formal methods
Not all of the faults in a software system are of equal importance.			1.1			
The design of systems can influence (and should minimise if possible) the effects of bugs.						
It is necessary to understand all of the possible conditions of use and applications of a computer-based system (or a part of a system) before you can test it completely.			1			
It is possible to test for all foreseeable failure modes of a computer-based system.			1.1[23]			
All failure modes of a software artifact can be predicted in advance.			1 - 3			van Vliet (1993, p 151)
Test plans should be prepared for software before it is written.			2			
Test plans should be prepared by designers rather than implementers.			- 1			QA approach
Systems should be tested by their eventual users before going live.			1			
• Beliefs Related to Quality						
The definition of 'quality' of a CBSD process or product is related to the circumstances under which the work is performed.						Argument in Chapter 6

22 M.A. Jackson, B.C.S. 1994 lecture referred to above.

23 Subject B's comments indicated that he believed that this element is part of the discipline 'wisdom' but that he is sceptical.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Controlling the CBSD process produces a better product.	2	3 - 1[24]	5			BS 5750
The process model used for CBSD will affect the quality of the product.			1	1		
A quality-aware[25] CBSD process will result in a higher quality product.	1	4	2			
CBSD process quality is defined entirely by the ability to generate high quality products.			1.1			
The organisational context (organisational goals, etc.) of a system's development will affect the definition of software quality.			2			
The structure of the team developing a software system has a significant effect in determining the system's quality.			1			Brooks (1975)
Software product standards are of importance in maintaining software quality.			3			Sommerville (1992, p 594)
Software process standards are of importance in maintaining software quality.	1	2.1	2			Sommerville (1992, p 594)
Software product standards are independent of software process standards.	1	2.1	3			
Software process and/or product standards should always be in force.			- 1			QA approach
Achieving software quality is made easier by simplifying the problem.			1			Simon (1975)
'Software quality' can be measured objectively.						
Software quality can be assessed by quantitative measurements.	1	3	1 - 1			
Software quality can *only* be assessed by quantitative measurements.			1			
The number of change requests and/or errors reported in software is a good measure of its quality.			1			
Quantitative measures of quality are better than qualitative measures.						

24 The dissenting voice is that of Booch.

25 Without defining 'quality-aware' too specifically here.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Not all software products have to be developed to a high level of quality.			1			
The cost of developing a software product is an important aspect of its quality: generally, cheaper is better.			1			
Software quality can be measured by comparing actual results with expected results.			1			

- Beliefs Related to the Computational Model of a Notation, Technique, Process or Method

Should the computational model of the notation used be . explicit, or . implicit?			1			
The similarities between the computational models of some notations allows families of notations to be identified using this criterion.			1			
The essence of a computer-based system can be captured in a finite number of types of models.				1		
This computational model or set of computational models (i.e. the one I'm using now) is most appropriate for the application in question.			1	1		
A modelling notation should be selected to match the problem.			1			Extension of Episkopou and Wood-Harper (1986)
The computational model of a notation should debar a process from presenting different results to the same question.						Birrell and Ould (1985, p 120)
The computational models of design notation and implementation language/mechanism should be well matched (or identical).			1			
The computational model of the selected implementation language should drive the choice of design notation computational model.			1			Informal observation of practice

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Metaphysical Beliefs About Software (Component) Reuse						
Software should be designed with reuse in mind.	1	3				
It is easier to use an existing solution than to re-solve the problem.						
The benefits of using reused designs outweigh the costs, at least to the extent that software reuse should be considered.			1			
Software reuse is a mandatory requirement.			1			
The benefits of using reused software can outweigh the disadvantages of doing so.			3			
Building up systems from reused components is a good thing at the personal level.			1			
Building up systems from reused components can produce a better quality system.			1			
Reused components will generally work to their specifications.			2			
It can be better to design round an existing component than to modify or refuse to use it.			1			
• Metaphysical Beliefs About Decomposition						
Reductionist vs. systemic approaches.						Checkland (1990)
The labels and concepts introduced at any level of abstraction should be defined at that level.						Requirement for definitions in notations
The same notation is valid at more than one level of abstraction.						Observation of definitions of notations
The same notation is valid at *all* levels of abstraction.			0.1			
The same notation is valid at all levels of abstraction down to that at which the next notation is used.						Derived from above
Top-down vs. bottom-up in principle: a continuum from top-down decomposition to bottom-up composition:						

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
. top-down decomposition is better than bottom-up composition.						Implicit in SD (Yourdon and Constantine, 1978), disagreed with by Jackson (1983)
. bottom-up composition is better than top-down decomposition.			1			
Top-down vs. bottom-up in problem definition: a continuum:						
. problems can be better understood by a process of breaking the complete problem into smaller bits.			2			Friedman with Cornford (1989, p 131)
. problems can be better understood by a process of building a model of the complete problem from smaller bits.			1			
Top-down vs. bottom-up in solution building: a continuum:						
. solutions can be better generated by a process of breaking the complete solution into smaller bits.	3	2 - 1[26]	4			
. solutions can be better generated by a process of building up the complete solution from smaller bits.			1			Smalltalk: Shafer and Ritz (1991, p 30)
Top-down decomposition requires knowledge of the subject area.			1			Jackson (1983)

- The Designer's Viewpoint - beliefs arising from the designer's world-view

'Tool' (craft) vs. representational (of an objective world) perspectives.						Spaul (1994)

26 "JSD is not top-down. This is a proud claim, not an embarrassed confession." (Jackson, 1983, p.370).

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
There will be a computer in there somewhere.			- 1			'Traditional' approach (Wood-Harper and Fitzgerald, 1982, p 13)
The computer-based 'system' is more important than the outside world.					- 1	Jackson[27]
One 'machine' or many?						Jackson[28]
Some design problems have been 'solved'.						Observation: e.g. payroll systems

- Management-Related Beliefs (often associated with Methods, but not necessarily so)

The software development process *is* capable of being managed.	2	4	3			Friedman with Cornford (1989)
Controlling the software process produces benefits which outweigh the costs.	1	4	1			
The costs and timescales of a computer-based system development can be estimated in advance with a reasonable degree of accuracy.	1	0.1 - 1	3 - 1			
The past is a good guide to project costs and timescales.			1			Special case of Principle of Uniformity of Systems
Using current project estimation techniques is better than not using them.			1			
The unit of granularity of project cost and/or timescale estimation is consistent from project to project.			1			
Project timescales can (almost) always be tightened.			1			
Certain qualitative measures can be identified by which the software development process can be managed and controlled.						Extension from other elements

27 M.A. Jackson, B.C.S. 1994 lecture referred to above.

28 M.A. Jackson, B.C.S. 1994 lecture referred to above.

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Certain quantitative measures can be identified by which the software process can be managed and controlled.						Birrell and Ould (1985)
Staff involved in the software process should be as productive as possible.			1 - 1[29]			
Tools used to support software development should have as a major objective the making of staff involved in the software process as productive as possible.						

- Beliefs Underlying Predefined Notations and their Use in Design

Predefined design notations are sufficiently mutable to allow an efficient search through the solution space (with, for some process models, backtracking if necessary).			- 1			
Physical differences between the graphical elements of different notations are (or are not!) significant.						Meditation on notations
Predefined notations are better than informal *ad hoc* notations.[30]			0.1 - 3[31]			
Predefined notations should be used for model building.			- 2			Implicit in predefined methods
A notation should be completely defined before work using it begins.			1 - 1			
Predefined modelling tools and notations allow more effective use of human analytical powers.			- 1			
The number of types of model required (values might be 'one', 'more than one', ...).						Observation of methods

---

29 The disagreement came from an individual developer working for himself.

30 Note that this might not be common to all computer-based systems developers; see subject C transcript, p 6.

31 This result is sufficiently interesting to lead me to quote from one of the disagreeing subjects: "I think that why I am cynical about them is that I think that sometimes it leads to, when you look at someone's analysis that has been broken in a formal way, it doesn't really seem to be saying very much. You get lots and lots of neat diagrams and it looks as if it's, er, all very wonderful but in a way it is saying things that are very very bland and, er, you know they aren't necessarily getting to the heart of the thing" (subject C, transcript p 6).

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
The primary use of formalised notations is for:						
. documentation.						
. as a design reasoning tool.			1			
Graphical tools are an aid to the mental processes of design.			1			
A (level of a) model can be defined in the head before being written down.			1			

- Beliefs Related to How to Research into Methodological Issues

Academic world exercises are sufficiently close to real-world exercises to allow valid conclusions to be drawn from one to the other.				1 - 1[32]		Action research
The lessons learned from small scale experiments can be scaled up to real world use of methods.				1		Action research

- Beliefs Related to How to Learn to Develop Computer-based Systems

Software development needs to be learned by practice or example in addition to theory classes.			1			Craft nature of CBSD
Experience with a method affects the way in which a method is used.			2	1		cf. Floyd (1983) and others using students as experimental subjects
Experience with an implementation mechanism affects the way in which that mechanism is used.			1.1			
Experience or knowledge of an organisation within which a system is developed affects the development of that system.			1			

32 Note the comment on performing the activity of action research (Avison and Wood-Harper, 1993, p 180): "A major strand of action research is that the practitioners [who? with what experience? what personal DM do they bring?] should participate in the analysis, design and implementation process and contribute at least as much as researchers in any decision-making. Thus there is synergy between the researchers and practitioners; the researchers building up theories and modifying them on the basis of practical experience and the practitioners using and modifying research ideas for solving real-world problems."

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
• Beliefs about Standards						
Different reasons for having standards:						
. control of the process/staff.						QA, quantitative measures of quality, Taylorism
. easing staff transfers between teams.			2			
. easier to recruit staff if you use something which is well-known in the industry.						Observation within practice
. easier to retain staff if you train them in something which is well-known in the industry.						Observation within practice
. aids maintainability if you keep to well-understood standards.			1			
. aids portability if you keep to well-understood standards.						
. aids project estimating.			1			
Personal loyalty to certain sets of standards.						
Intra-organisation standards are important.			1			
Intra-organisation standards are more important than inter-organisation standards.						Observation of market
Aspects of standards can be recognised (and are a good thing) even if applied informally.			1			
2.3.3 Values						
Pragmatism.			1			
Computer-based systems should (or should not necessarily) be socially useful.						'Scandinavian school'
Independence of specific computer-based tools is *not* important when designing notations, etc.						Birrell and Ould (1985), Downs *et al.* (1992)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
Independence of specific aspects of the development, such as the implementation language, is *not* important when designing notations, etc.	1	1				
The system (and/or its data) is more important than the wishes of its users.			1			
People interacting with a system can be treated as objects, and reduced to (impersonal) roles.	-1	-1.2	1 - 0.1			Bansler and Bodker (1993)
The actual users of a system are less important than their managers.			1			
Political correctness.			1			cf. fashion
Novelty in a CBSD method etc.; *newer* is *better*						Argument in Chapter 6; cf. fashion - but see Blum (1993, p 126)
Sequential operation (vs. parallel):						
. within a module.	1	1				
. between modules.						
Values defining a project's 'success':						
. meeting the specification.			1			
. exceeding the specification.						Personal experience
. long term maintainability.						
. it works.			1			
. it sells.			2			
. it makes a profit.			3			
. it gets launched on time.			2 - 1			
. it gets used.			2			
. it makes the end users happy.			2			
. it gets us the next tranche of funding (academic success).						my cynicism!
Aesthetics in a system's externals.	0.1	0.1 - 1				Vitruvius (1931)

Element	*Bk1*	*Bk2*	*Intvw*	*Act'n*	*Fail*	*Other*
2.3.4 Exemplars						
Exemplars of software products.						
Exemplars of experience:						
. the Fortran compilers.			1			
. the rounding fraud.			1			Russel Winder (personal communicat'n)
. 'computer generated people'.						Equity Funding Corporation of America
. the U.S. army payroll syndrome.						anecdotal
Specific examples included in the literature on tools, techniques and methods.						Jackson (1983)
Exemplars of CBSD process models:						
code-and-fix.						Boehm *et al.* (1984)
specifying.						Boehm *et al.* (1984)
prototyping.						Boehm *et al.* (1984)
spiral.						Boehm (1988)
Operational research.						
HCI - the Xerox Star user interface.						
Design plans in use in CBSD						

Appendix 3: Glossary

The following abbreviations have been used in this thesis:

CBSD	computer-based systems development; the discipline whose state and problems are addressed by this thesis
DM	disciplinary matrix; the Kuhnian 'paradigm' in the sense of the "constellation of group commitments" (Kuhn, 1970, p 181) shared by the members of a scientific community
IS	Information Systems
MIS	Management Information Systems

References

Avison D.E. and Wood-Harper A.T. (1993) *Multiview: An Exploration in Information Systems Development*; Alfred Waller; Henley-on-Thames (originally published 1990; 1993 reprint).

Bansler J.P. and Boedker K. (1993) "A reappraisal of Structured Analysis: design in an organisational context"; *ACM Trans. on Information Systems*; 11 (2); 165–193.

Banville C. and Landry M. (1989) "Can the field of MIS be disciplined?"; *Comm. ACM*; 32 (1); 48–60.

Beeson I. (1994) "The body in the information system"; *delivered at* Philosophical and Logical Aspects of Information Systems Symposium, University of the West of England, Bristol, 20-22 April, 1994 [original transcript as distributed to participants].

Benyon D. and Skidmore S. (1987), "Towards a tool kit for the systems analyst"; *Computer Journal*; 30 (1); 2–7.

Birrell N.D. and Ould M.A. (1985) *A Practical Handbook for Software Development*; Cambridge University Press; Cambridge.

Blum B.I. (1993) "The economics of Adaptive Design"; *Journal of Systems and Software*; 21 (2); 117–128.

Blum B.I. (1994) "Characterizing the software process"; *Information and Decision Technologies*; 19 (4); 215–232.

Blum B.I. (1995) "Resolving the software maintenance paradox"; *Software Maintenance: Research and Practice*; 7; 3–26.

Boehm B.W. (1981) *Software Engineering Economics*; Prentice-Hall; cited in van Vliet (1993, p 103 *et seq.*)

Boehm B.W, Gray T.E. and Seewaldt T. (1984) "Prototyping versus specifying: a multiproject experiment"; *IEEE Transactions on Software Engineering*; SE–10 (3); 290–302.

Boehm B.W. (1988) "A Spiral Model of software development and enhancement"; *IEEE Computer*; 21 (5); 61–72.

Boland R.J. (1987) "The in-formation of information systems"; in Boland R.J. and Hirschheim R.A. (Eds.); *Critical Issues in Information Systems Research*; Wiley; *cited by* Beeson (1994).

Booch G. (1994) "The Booch Method: process and pragmatics"; in Carmichael (1994); pp 149–166.

Brooks F.P. (1975) *The Mythical Man-Month*; Addison-Wesley; Reading, MA.

Brooks F.P. (1987) "No silver bullet"; *IEEE Computer*; 20 (4); April 1987; 10–19.

Burrell G. and Morgan G. (1979) *Sociological Paradigms and Organizational Analysis*; Heinemann; London; *cited in* Hirschheim and Klein (1989).

Carmichael A. (Ed.) (1994) *Object Development Methods*; SIGS Books; New York

Chalmers A.F. (1992) *What is This Thing Called Science?*; Second Edition; Open University Press; Buckingham (1982 – 1992 reprint).

Checkland P. (1990) *Systems Thinking, Systems Practice*; Wiley; Chichester; reprint of 1984 revised edition.

Coad P. and Yourdon E. (1991) *Object-Oriented Analysis*; Second Edition; Prentice-Hall International, Inc.; Englewood Cliffs, NJ.

Connaughton C. and Dampney C.N.G. (1994) "Surviving paradigm shifts in software development technology: a management case study of industrial experience"; *Australian Computer Journal*; 26 (4); 114–123.

Constantine L.L. and Yourdon E. (1979); *Structured Design*; Prentice-Hall; Englewood Cliffs, NJ; *cited in* Sommerville (1992).

Constantine L.L. (1993) "Work organization: paradigms for project management and organization"; *Comm. ACM*; 36 (10); 35–43.

Curtis B., Krasner H. and Iscoe N. (1988) "A field study of the software design process for large systems"; *Comm. ACM*; 31 (11); 1268–1287.

Curtis B. (1990) "Empirical studies of the software design process"; in D. Diaper *et al.* (Eds.); *Proc. INTERACT 90*; Elsevier Science Publishers B.V.; pp xxxv–xl.

Dahlbom B. and Mathiassen L. (1993) *Computers in Context*; NCC Blackwell; Cambridge, Mass.

Dasgupta S. (1991) *Design Theory in Computer Science*; Cambridge University Press; Cambridge.

Dix A., Finlay J., Abowd G. and Beale R. (1993) *Human–computer Interaction*; Prentice-Hall; New York.

Dowell J. and Long J. (1989) "Towards a conception for an engineering discipline of human factors"; *Ergonomics*; 32 (11); 1513–1535.

Downs E., Clare P. and Coe I. (1992) *Structured Systems Analysis and Design Method*; Second Edition; Prentice Hall; New York.

Episkopou D.M. and Wood-Harper A.T. (1986) "Towards a framework to choose appropriate IS approaches"; *The Computer Journal*; 29 (3); 222–228.

Essink J.B. (1983) "A modelling approach to information system development" in Olle *et al.* (1983); pp 55–86.

Farhoomand A.F. (1987) "Scientific progress of management information systems"; *Data Base*; 18 (2); 48–56.

Fenton N.E. (1991) *Software Metrics*; Chapman & Hall; London.

Feyerabend P. (1975) *Against Method*; NLB; London.

Feyerabend P. (1993) *Against Method*; Third Edition; Verso; London.

Fisher P. (1994) "Just not the TickIT", *Computer Guardian*; in *The Guardian*; 31 March 1994; p 19.

Flowers S. (1994) "One huge crash"; *Computer Guardian*; in *The Guardian*; 28 April 1994; p 21.

Floyd C. (1983) "A comparative evaluation of system development methods"; in Olle *et al.* (1983)

Friedman A.L. with Cornford D.S. (1989) *Computer Systems Development: History, Organisation and Implementation*; Wiley; Chichester.

Funk K., Lyall B. and Riley V. (1995) "Flightdeck automation problems"; to appear in Jensen R.S. (Ed.), *Proceedings of the Eighth International Symposium on Aviation Psychology*; 24–27 April 1995; Columbus, Ohio.

Gasson S. (1994) "Managing organisational change: the impact of information system development methods"; in Lissoni *et al.* (1994); pp 139–150

Gilb T. (1985) "Evolutionary delivery versus the 'Waterfall Model'"; *ACM Sigsoft Software Engineering Notes;* 10 (3); 49–61

Gordon J.E. (1978) *Structures, or Why Things Don't Fall Down*; Penguin Books; Harmondsworth.

Green T.R.G. (1989) Cognitive dimensions of notations; in Sutcliffe and Macaulay (1989); pp 443–460.

Gregory F.H. (1994) "Mapping information systems on to the real world"; *delivered at* Philosophical and Logical Aspects of Information Systems Symposium, University of the West of England, Bristol, 20-22 April, 1994 [original transcript as distributed to participants].

Groenbaek K., Kyng M. and Mogensen P. (1993) "CSCW challenges: cooperative design in engineering projects"; *Comm. ACM*; 36 (4); 67–77.

Gutting G. (Ed.) (1980) *Paradigms & Revolutions*; University of Notre Dame Press; Notre Dame.

Hirschheim R. and Klein H.K. (1989) "Four paradigms of information systems Development", *Comm. ACM*; 32 (10); 1199–1216.

Hospers J. (1970) *An Introduction to Philosophical Analysis*; Routledge and Kegan Paul Ltd, London; 1967 (1970 reprint).

Jackson M.A (1983) *System Development*; Prentice Hall; Englewood Cliffs, NJ.

Jackson M.A. (1989) "Methodology for the '90s"; *Microprocessing and Microprogramming*; 27; 7–16.

Jackson M.C. and Keys P. (1984) "Towards a system of systems methodologies"; *J. Operational Research Society*; 35 (6); 473–486.

Jayaratna N., Paton G, Merali Y. and Gregory F. (Eds.) (1993) *Proc. Conference on the Theory Use and Integrative Aspects of IS Methodologies, 1–3 September 1993*; British Computer Society Information Systems Methodologies Specialist Group.

Junk W. and Oman P. (1992) The influence of software engineering paradigms on individual and team project results; in Sledge (1992); pp 417–436.

Klein H.K. and Hirschheim R.A. (1987) "A comparative framework of data modelling paradigms and approaches"; *Computer Journal*; 30 (1); 8–15.

Kneller G.F. (1978) *Science as a Human Endeavour*; Columbia University Press; New York.

Kuhn T.S. (1962) *The Structure of Scientific Revolutions*; University of Chicago Press; Chicago.

Kuhn T.S. (1970) *The Structure of Scientific Revolutions*; Second Edition, Enlarged; University of Chicago Press; Chicago.

Kuhn T.S. (1977) "Second thoughts on paradigms"; in T.S. Kuhn, *The Essential Tension*; University of Chicago Press; Chicago; pp 293–319.

LAS (1993) *Report of the Inquiry into the London Ambulance Service*; South West Thames Regional Health Authority; London.

Liang T.Y. (1994) "The Basic Entity Model: a fundamental theoretical model of information and information processing"; *Information Processing and Management*; 30 (5); 647–661.

Licker P.S. (1987) *Fundamentals of Systems Analysis with Application Design*; Boyd & Fraser; Boston.

Lissoni C., Richardson T., Miles R., Wood-Harper T. and Jayaratna N. (Eds.) (1994) *Information Systems Methodologies 1994: Second Conference on Information Systems Methodologies, 31 August–2 September 1994;* British Computer Society Information Systems Methodologies Specialist Group.

Long J. and Dowell J. (1989) "Conceptions of the discipline of HCI – craft, applied science and engineering"; in Sutcliffe and Macauley (1989).

Maguire S. (1994) "A new philosophy, a new agenda: introducing information systems into complex organisations"; *delivered at* Philosophical and Logical Aspects of Information Systems Symposium, University of the West of England, Bristol, 20-22 April, 1994 [original transcript as distributed to participants].

Makepeace R. (1993) *Methodological Responses to 'The Software Crisis': A Philosophical View*; dissertation, Staffordshire University.

Martin B. (Ed.) (1994) *Price Waterhouse Information Technology Review 1994/5*; Price Waterhouse; London.

Miller G.A. (1956) "The magical number seven, plus or minus two: some limits on our capacity for processing information"; *Psychological Reviews*; 63; 81–97; cited by van Vliet (1993).

Midgley G. (1989) "Critical systems and the problem of pluralism"; *Cybernetics and Systems*; 20; 219–231.

Mumford E. (1993) "The ETHICS approach"; *Comm. ACM*; 36 (4); 82 (a short summary of the method and its user-oriented philosophy)

Musgrave A.E. (1971) "Kuhn's second thoughts"; *British Journal for the Philosophy of Science*; 22; 287–297. Reprinted in Gutting (1980); pp 39–53, from which the page numbers are taken.

Myers G.J. (1975) *Reliable Software through Composite Design*; Van Nostrand Reinhold; New York; cited in Birrell and Ould (1985).

Myers G.J. (1978) *Composite/Structured Design*; Van Nostrand Reinhold; New York; cited by Schach (1993, pp 240–254) as 1978b.

Naur P. and Randell B. (1969) *Software Engineering; Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7 – 11 October 1968*; Scientific Affairs Division, NATO; Brussels.

Newton I. (1726) *Principia Mathematica*; Third Edition; translated by A. Motte (1729), revised F. Cajori (1934); University of California Press; Berkeley, CA.

Olle T.W., Sol H.G. and Tully C.J. (Eds.) (1983) *Information Systems Design Methodologies: A Feature Analysis*; Elsevier.

Petre M. (1989) *Finding a Basis for Matching Programming Languages to Programming Tasks*; PhD Thesis; University College London; London.

Petre M. (1995) "Why looking isn't always seeing"; *Comm. ACM*; 38 (6); 33–44.

Popper K. (1969) *Conjectures and Refutations*; Routledge and Kegan Paul; London.

Ralston A. and Reilly E.D. (1993) *Encyclopedia of Computer Science*; Chapman and Hall; London.

Rensch T. (1982); "Object-oriented programming"; *SIGPLAN Notices*; 17 (9); 51–57; quoted by van Vliet (1993).

Robb F. (1994) "Some philosophical and logical aspects of information systems", Opening address, Philosophical and Logical Aspects of Information Systems Symposium, University of the West of England, Bristol, 20-22 April, 1994, pp 2–3 [of original transcript as distributed to participants].

Royce W.W. (1970) "Managing the development of large software systems: concepts and techniques"; *Proc. IEEE WESCON*; pp 1–9; cited in van Vliet (1993).

Rumbaugh J., Blaha M., Premerlani W., Eddy F. and Lorensen W. (1991) *Object-Oriented Modelling and Design*; Prentice-Hall; Englewood Cliffs, NJ.

Schach S.R. (1993) *Software Engineering*; Richard D. Irwin, Inc., and Aksen Associates, Inc.; Homewood, Ill.

Shafer D. and Ritz D.A. (1991) *Practical Smalltalk*; Springer-Verlag; New York

Shapire D. (1964) "The Structure of Scientific Revolutions"; *Philosophical Review*; 73; pp 383–394. Reprinted in Gutting (1980); pp 27–38, from which the page numbers are taken – a review of the *first* edition of Kuhn's book.

Shepherd J., Morton A.H. and Spence L.F. (1991) *Higher Electrical Engineering*; Longman Scientific and Technical; Harlow, Essex; Second Edition (reprint of 1977 text).

Simon H.A. (1975) "Style in design"; in C.A. Eastman (Ed.); *Spatial Synthesis in Computer-Aided Building Design*; Applied Science Publishers; London.

Simpson H. (1986) "The Mascot method"; *Software Engineering Journal*; 1 (3); 103–120.

Sledge C. (Ed.) (1992) *Proc. Software Engineering Education: SEI Conference 1992*; Lecture Notes in Computer Science 640; Springer-Verlag.

Sommerville I. (1992) *Software Engineering*; Fourth Edition; Addison-Wesley; Wokingham.

Spaul M.W.J. (1994) "The tool perspective on information systems design: what Heidegger's philosophy can't do"; *delivered at* Philosophical and Logical Aspects of Information Systems Symposium, University of the West of England, Bristol, 20-22 April, 1994 [original transcript as distributed to participants].

Sutcliffe A. and Macaulay L. (Eds.) (1989) *People and Computers V: Proceedings CHI '89*; Cambridge University Press.

van Gigch J.P. and Pipino L.L. (1986) "In search of a paradigm for the discipline of information systems"; *Future Computer Systems*; 1 (1); 71–97.

van Vliet H. (1993) *Software Engineering: Principles and Practice*; Wiley; Chichester.

Vitruvius (1931) *De Architectura* (translated by F. Granger); Loeb edition; Harvard University Press; Cambridge, Mass.

Weinberg G. (1971) *The Psychology of Computer Programming*; Van Nostrand Reinhold; New York; 1971.

Wernick P. and Winder R. (1993) *A Plethora of Paradigms*; Research Note 93/14; Department of Computer Science; University College London; London.

Winder R., Sasse M.A. and Holti R. (1992) Collective Learning: a new conceptual framework for systems development; Research Note 92/88; Department of Computer Science; University College London; London .

Winder R. and Wernick P. (1993), "The inductive nature of software engineering and its consequences"; in Jayaratna *et al* (1993); pp 431–443.

Winder R. and Wernick P. (1994) "Refining a philosophical model of software development: tracing elements of the disciplinary matrix"; in Lissoni *et al* (1994); pp 203–216.

Winder R., Probert S. and Beeson I. (Eds.) (1996) *Philosophical Aspects of Information Systems*; UCL Press; London (in preparation).

Wood-Harper A.T. and Fitzgerald G. (1982), "A taxonomy of approaches to systems analysis"; *Computer Journal*; 25 (1); 12–16.

Yourdon E. and Constantine L.L. (1978), *Structured Design*; Prentice-Hall; Englewood Cliffs, N.J.; cited by Birrell and Ould (1985).

Yourdon E. (1989) *Modern Structured Analysis*; Yourdon Press/Prentice-Hall; New York *cited by* Bansler and Boedker (1993).

Zachman J.A. (1987) "A framework for information systems architecture"; *IBM Systems Journal*; 26 (3); 276–292.